

19

9

15

HEAP INTERACTIV

MANUALUL PROFESORULUI

Heap Lecții pentru **AEL** Built to teach *intelli gently*

Combinarea a două heap-uri

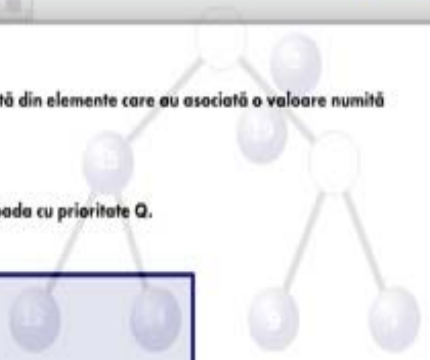
O astfel de operație creează la fiecare pas i heap-ul cu rădăcina $H[i]$, combinând două heap-uri de dimensiuni apropiate, heap-ul cu rădăcina $2 \cdot i$ cu heap-ul cu rădăcina $2 \cdot i + 1$ și cu elementul $H[i]$. Nodul tată (inițial i) este "retrogradat" fiind înlocuit cu fiul cu valoarea cea...

Heap Lecții pentru **AEL** Built to teach *intelli gently*

Coadă cu prioritate

Definiție
O coadă cu prioritate este o structură de date abstractă formată din elemente care au asociată o valoare numită cheie sau prioritate și care suportă următoarele operații:

- **Insert(Q,x):** inserează elementul x în coada cu prioritate Q ;
- **ExtractMax(Q):** extrage elementul de valoare maximă din coada cu prioritate Q .



Control animație

Rulează animație Complexitate

```

Heap (int i, int n)
{
    fiu, tata, aux;
    << i; fiu <- 2*i;
    timp fiu <- n
    while (fiu+1 <= n atunci
        daca H[fiu] < H[fiu+1] fiu <- fiu+1;
        daca H[tata] < H[fiu] atunci
        {
            aux <- H[tata];
            H[tata] <- H[fiu];
            H[fiu] <- aux;
            tata <- fiu; fiu <- fiu*2;
        }
    altfel fiu <- n+1;
}
    
```

Control animație

Rulează animație Complexitate

Obiective

Produs realizat de

prof. Emanuela Cerchez
 prof. Marinela Șerban
 Bianca Milatinovici
 Răzvan Grădinaru

Cuprins

1. Terminologie

2. Structură generală


- 2.1. Obiective didactice
- 2.2. Conținut
- 2.3. Recomandări de structurare și predare


3. Obiecte de conținut - detalieri


- 3.1. **M_{1,1}** – Noțiuni fundamentale – recapitulare
- 3.2. **M_{1,2}** – MaxHeap – prezentare
- 3.3. **M_{1,3}** – MinHeap – prezentare
- 3.4. **M_{1,4}** – Test
- 3.5. **M_{2,1}** – Inserarea unui nod într-un heap
- 3.6. **M_{2,2}** – Crearea unui nod prin inserări repetate
- 3.7. **M_{3,1}** – Combinarea a două heap-uri
- 3.8. **M_{3,2}** – Crearea unui heap prin combinare
- 3.9. **M_{4,1}** – Extragerea unui nod dintr-un heap
- 3.10. **M_{4,2}** – Coadă cu prioritate
- 3.11. **M_{4,3}** – Test recapitulativ coada cu prioritate
- 3.12. **M_{5,1}** – Heapsort
- 3.13. **M_{5,2}** – Test recapitulativ Heapsort

4. Bibliografie





1. Terminologie


Butoane definiție / explicații suplimentare –  – sunt amplasate în funcție de context și, atunci când sunt accesate, prezintă într-o fereastră de detaliu, definiția termenului respectiv sau explicații suplimentare despre acesta.


Butoane de reinițializare a animației / aplicației -  - prin apăsarea lor se reinițializează animația, respectiv aplicația.

Butoane care indică obiectivele lecției respective -  - sunt amplasate totdeauna în partea din dreapta jos a ecranului. Prin apăsarea lor, într-o fereastră detaliu se prezintă obiectivele lecției.



Butoane de control a animației -  - prin apăsarea butoanelor corespunzătoare:

- se execută animația “pas cu pas” – 
- se rulează animația în mod continuu – 
- se face pauză în executarea animației – 
- se oprește animația – 

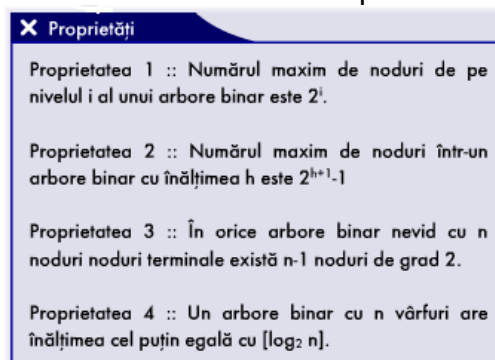
Butoane care indică anumite acțiuni care trebuie executate asupra elementului selectat –  – atunci când sunt accesate realizează operația indicată.

Butoane selectare item –  – atunci când sunt accesate se realizează operația de selectare și afișare a item-ului corespunzător din test.

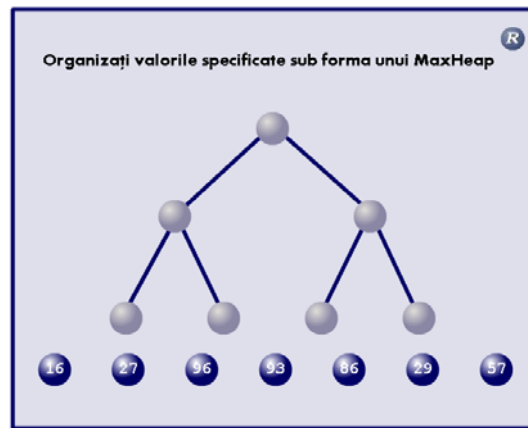
Butoane de indicare a corectitudinii răspunsului

- răspuns corect – 
- răspuns eronat – 

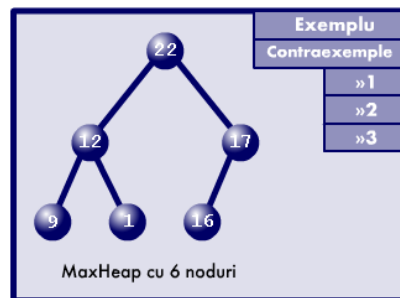
Ferestre detaliu – sunt ferestre care oferă informații suplimentare despre o anumită noțiune. Asupra unei ferestre detaliu se poate face "drag_and_drop" acționând asupra barei de titlu a ferestrei. Exemplu :



Ferestre de animație realizată cu mouse-ul – sunt ferestre în care animația este realizată direct cu mouse-ul, utilizând “drag_and_drop” asupra unui element. Exemplu :



Ferestre afișare contraexemple – sunt ferestre în care exemplul corect și contraexemplele sunt vizualizate prin selectare. Exemplu :



Butoane de selectare a modului de afișare al algoritmului – – atunci când sunt accesate realizează afișarea algoritmului în pseudocod sau în limbajul de programare indicat.

Butoane pentru închis ferestre detaliu – – sunt amplasate în colțul stânga sus a ferestrelor de detaliu iar apăsarea lor duce la închiderea ferestrei.

2. Structura generală

În acest capitol sunt prezentate obiectivele didactice care pot fi atinse utilizând acest material. În finalul prezentării sunt incluse câteva recomandări privind unele moduri în care ar putea fi combinate aceste momente pentru a obține o lecție.

2.1. Obiective didactice

Obiectiv	Detaliere
Obiective de referință	
R1	Analizarea modului de funcționare a heap-urilor
R2	Realizarea aplicațiilor utilizând algoritmi specifici
R3	Urmărirea etapelor de realizare a unei aplicații
Obiective operaționale	
OP1	Definirea corectă a noțiunilor de arbore cu rădăcină, înălțime, nivel
OP2	Definirea corectă a noțiunii de arbore binar
OP3	Definirea corectă a noțiunii de arbore binar plin
OP4	Definirea corectă a noțiunii de arbore binar complet
OP5	Recunoașterea unui arbore (cu rădăcină, binar, binar plin, binar complet)
OP6	Construirea unui arbore (binar, binar plin, binar complet)
OP7	Reprezentarea unui arbore binar complet
OP8	Definirea unui MaxHeap (MinHeap)
OP9	Inserarea unui nod într-un heap
OP10	Crearea unui heap prin inserări repetate
OP11	Urmărirea execuției “pas cu pas” a unui algoritm
OP12	Implementarea unui algoritm într-un limbaj de programare
OP13	Determinarea complexității timp a unui algoritm
OP14	Combinarea a două heap-uri
OP15	Extragerea valorii maxime dintr-un heap
OP15	Definirea corectă a noțiunilor de coadă, coadă cu prioritate
OP16	Recunoașterea situațiilor când este necesară coada cu prioritate
OP17	Identificarea modalităților de reprezentare a unei cozi cu prioritate
OP18	Descrierea metodei de sortare HeapSort
OP19	Dezvoltarea gândirii algoritmice, logice, flexibile, creatoare
OP20	Dezvoltarea atenției concentrate și spiritului de observație

2.2 Conținut

Se prezintă lista obiectelor de conținut (notate cu M) și caracteristicile lor generale.

M_{1.1} – Noțiuni fundamentale – recapitulare	
Obiective didactice	OP1, OP2, OP3, OP4, OP5, OP6, OP7, OP19, OP20
Timp de predare	30 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, algoritmizare, studiu de caz • metode de acțiune: exercițiul, învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală, exercițiul de consolidare
Descriere	<ul style="list-style-type: none"> • prezentarea proprietăților diferitelor tipuri de arbori • exemplificarea diferitelor tipuri de arbori folosind reprezentarea grafică • construcția diferitelor tipuri de arbori folosind animația • prezentarea proprietăților diferitelor tipuri de arbori • testarea cunoașterii noțiunilor prezentate
Cuvinte cheie	<ul style="list-style-type: none"> • arbore cu rădăcină • rădăcina arborelui • arbore binar • arbore binar plin • arbore binar complet • nivel al unui nod • înălțimea unui arbore • tatăl unui nod • fiul stâng al nodului • fiul drept al nodului

M_{1.2} – MaxHeap – prezentare	
Obiective didactice	OP4, OP6, OP7, OP8, OP19, OP20
Timp de predare	10 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, studiu de caz • metode de acțiune: exercițiul, învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală, exercițiul de consolidare
Descriere	<ul style="list-style-type: none"> • prezentarea noțiunii de MaxHeap • prezentarea unor contraexemple • exersarea construirii unui MaxHeap
Cuvinte cheie	<ul style="list-style-type: none"> • maxheap

M_{1.3} – MinHeap – prezentare	
Obiective didactice	OP4, OP6, OP7, OP8, OP19, OP20
Timp de predare	5 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, studiu de caz • metode de acțiune: exercițiul, învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală, exercițiul de consolidare
Descriere	<ul style="list-style-type: none"> • prezentarea noțiunii de MinHeap • prezentarea unor contraexemple • exersarea construirii unui MinHeap
Cuvinte cheie	<ul style="list-style-type: none"> • minheap

M_{1.4} – Test	
Obiective didactice	OP4, OP5, OP6, OP7, OP8
Timp de predare	5 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • evaluare în formă scrisă prin intermediul calculatorului
Descriere	<ul style="list-style-type: none"> • test grilă
Cuvinte cheie	<ul style="list-style-type: none"> • maxheap, minheap

M_{2.1} – Inserarea unui nod într-un heap	
Obiective didactice	OP7, OP9, OP11, OP12, OP13, OP19, OP20
Timp de predare	30 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, algoritmizare, studiu de caz • metode de acțiune: învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • prezentarea modalității de inserare a unui nou nod • prezentarea animației de inserare a unui nou nod "pas cu pas" • prezentarea algoritmului de inserare a unui nou nod în pseudocod, Pascal sau C++ • descrierea determinării complexității timp a algoritmului de inserare a unui nou nod
Cuvinte cheie	<ul style="list-style-type: none"> • heap • inserare nod • complexitate timp

M_{2.2} – Crearea unui heap prin inserări repetate	
Obiective didactice	OP7, OP8, OP9, OP10, OP11, OP12, OP13
Timp de predare	20 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, algoritmizare • metode de acțiune: exercițiul, învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • prezentarea modalității de inserare a unui nou nod • prezentarea modalității de creare a unui heap prin inserări repetate • prezentarea algoritmului de inserare a unui nou nod în pseudocod, Pascal sau C++ • descrierea determinării complexității timp a algoritmului de inserare a unui nou nod
Cuvinte cheie	<ul style="list-style-type: none"> • heap • inserare nod • complexitate timp

M_{3.1} – Combinarea a două heap-uri	
Obiective didactice	OP8, OP11, OP12, OP13, OP14, OP19, OP20
Timp de predare	30 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, algoritmizare, studiu de caz • metode de acțiune: învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • prezentarea modalității de combinare a două heap-uri cu un nou nod • prezentarea "pas cu pas" a animației de combinare a două heap-uri cu un nou nod • prezentarea algoritmului de combinare a două heap-uri cu un nou nod în pseudocod, Pascal sau C++ • descrierea determinării complexității timp a algoritmului de combinare a două heap-uri cu un nou nod
Cuvinte cheie	<ul style="list-style-type: none"> • heap • combinare heap-uri cu un nod • complexitate timp

M_{3.2} – Crearea unui heap prin combinare	
Obiective didactice	OP8, OP11, OP12, OP13, OP14
Timp de predare	20 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, algoritmizare • metode de acțiune: exercițiul, învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • prezentarea modalității de creare a unui heap prin combinări repetate • prezentarea algoritmului de creare a unui heap prin combinare în pseudocod, Pascal sau C++ • descrierea determinării complexității timp a algoritmului de creare a unui heap prin combinare
Cuvinte cheie	<ul style="list-style-type: none"> • heap • combinare heap-uri • complexitate timp

M_{4.1} – Extragerea unui nod dintr-un heap	
Obiective didactice	OP8, OP11, OP12, OP13, OP15
Timp de predare	25 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație, algoritmizare • metode de acțiune: exercițiul, învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • prezentarea modalității de extragere dintr-un heap a nodului cu valoarea maximă • prezentarea "pas cu pas" a animației de extragere dintr-un heap a nodului cu valoarea maximă • prezentarea algoritmului de extragere dintr-un heap a nodului cu valoarea maximă în pseudocod, Pascal sau C++ • descrierea determinării complexității timp a algoritmului de extragere dintr-un heap a nodului cu valoarea maximă
Cuvinte cheie	<ul style="list-style-type: none"> • heap • extragere nod • complexitate timp

M_{4.2} – Coada cu prioritate	
Obiective didactice	OP15, OP16, OP17
Timp de predare	15 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație • metode de acțiune: învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • definirea unei cozi cu prioritate • prezentarea unei situații reale care necesită coada cu prioritate – utilizarea unei imprimante de rețea • prezentarea "pas cu pas" a animației pentru situația reală dată • studierea comparativă din punctul de vedere a complexității timp a câtorva metode de implementare a cozilor cu prioritate
Cuvinte cheie	<ul style="list-style-type: none"> • coadă • coadă cu prioritate • complexitate timp

M_{4,3} – Test recapitulativ coada cu prioritate	
Obiective didactice	OP15, OP16, OP17
Timp de predare	10 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • evaluare în formă scrisă prin intermediul calculatorului
Descriere	<ul style="list-style-type: none"> • test grilă
Cuvinte cheie	<ul style="list-style-type: none"> • coadă • coadă cu prioritate • complexitate timp

M_{5,1} – HeapSort	
Obiective didactice	OP18, OP19, OP20
Timp de predare	35 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none"> • metode de comunicare orală: expunere, conversație • metode de acțiune: învățarea prin descoperire • procedee de instruire: explicația în etapa de comunicare; învățarea prin descoperire dirijată, inductivă, experimentală
Descriere	<ul style="list-style-type: none"> • descrierea metodei de sortare heapsort • prezentarea "pas cu pas" a animației ce prezintă metoda de sortare heapsort • prezentarea algoritmului de sortare heapsort în pseudocod, Pascal sau C++ • studierea comparativă din punctul de vedere a complexității timp a câtorva metode de sortare
Cuvinte cheie	<ul style="list-style-type: none"> • sortare • sortare prin numărare • metoda bulelor • sortare prin selecție • sortare prin inserție • sortare prin interclasare • sortare rapidă • heapsort • complexitate timp

M_{5,2} – Test recapitulativ HeapSort	
Obiective didactice	OP18
Timp de predare	15 min
Tip de interacțiune cu elevii	<ul style="list-style-type: none">• evaluare în formă scrisă prin intermediul calculatorului
Descriere	<ul style="list-style-type: none">• test grilă
Cuvinte cheie	<ul style="list-style-type: none">• sortare• minheap• heapsort• sortare prin numărare• metoda bulelor• sortare prin selecție• sortare prin inserție• sortare prin interclasare• sortare rapidă• complexitate timp

2.3. Recomandări de structurare și predare

- **Planul unității de învățare 1** **Timp: 1 oră**

Obiect de conținut	Timp (min)
M _{1,1}	30
M _{1,2}	10
M _{1,3}	5
M _{1,4}	5

- **Planul unității de învățare 2** **Timp: 1 oră**

Obiect de conținut	Timp (min)
M _{2,1}	30
M _{2,2}	20

- **Planul unității de învățare 3** **Timp: 1 oră**

Obiect de conținut	Timp (min)
M _{3,1}	30
M _{3,2}	20

- **Planul unității de învățare 4** **Timp: 1 oră**

Obiect de conținut	Timp (min)
M _{4,1}	25
M _{4,2}	15
M _{4,3}	10

- **Planul unității de învățare 5** **Timp: 1 oră**

Obiect de conținut	Timp (min)
M _{5,1}	35
M _{5,2}	15

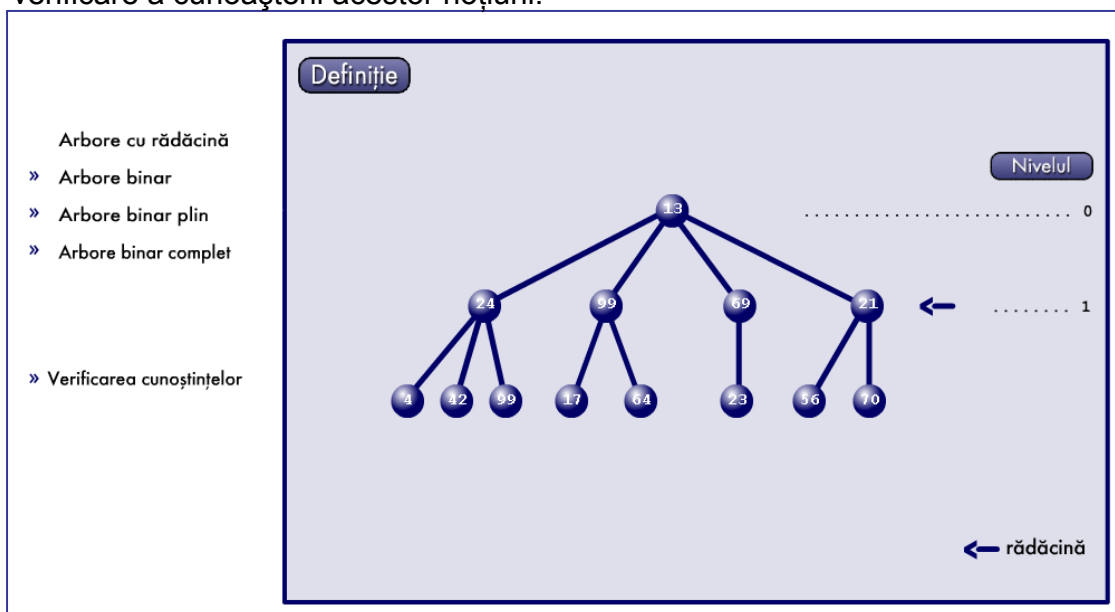
3. Obiecte de conținut - detalieri

În continuare vom prezenta în detaliu modul de utilizare a elementelor din ferestrele lecției (navigare, elemente specifice, funcționarea aplicațiilor, etc.). Subliniem că navigarea elementară se face cu ajutorul butoanelor descrise în Capitolul 1 – Terminologie, al acestui manual. Nu ne vom referi la acestea decât spicuitiv.

3.1. Noțiuni fundamentale – recapitulare

În acest obiect de conținut sunt prezentate noțiunile recapitulative necesare pentru studierea structurii de date heap.

În partea stângă a ecranului apare un meniu cu cinci itemi, precedați de butonul » care se animează în momentul în care cursorul mouse-ului trece peste item. Obiectul de conținut cuprinde patru noțiuni recapitulative – arbore cu rădăcină, arbore binar, arbore binar plin, arbore binar complet – și un test recapitativ de verificare a cunoașterii acestor noțiuni.



La acționarea primului item – **Arbore cu rădăcină** – începe automat animația care construiește pas cu pas un arbore oarecare indicându-se în fiecare moment rădăcina curentă. Pe tot parcursul construirii arborelui sunt disponibile două butoane :

Definiție

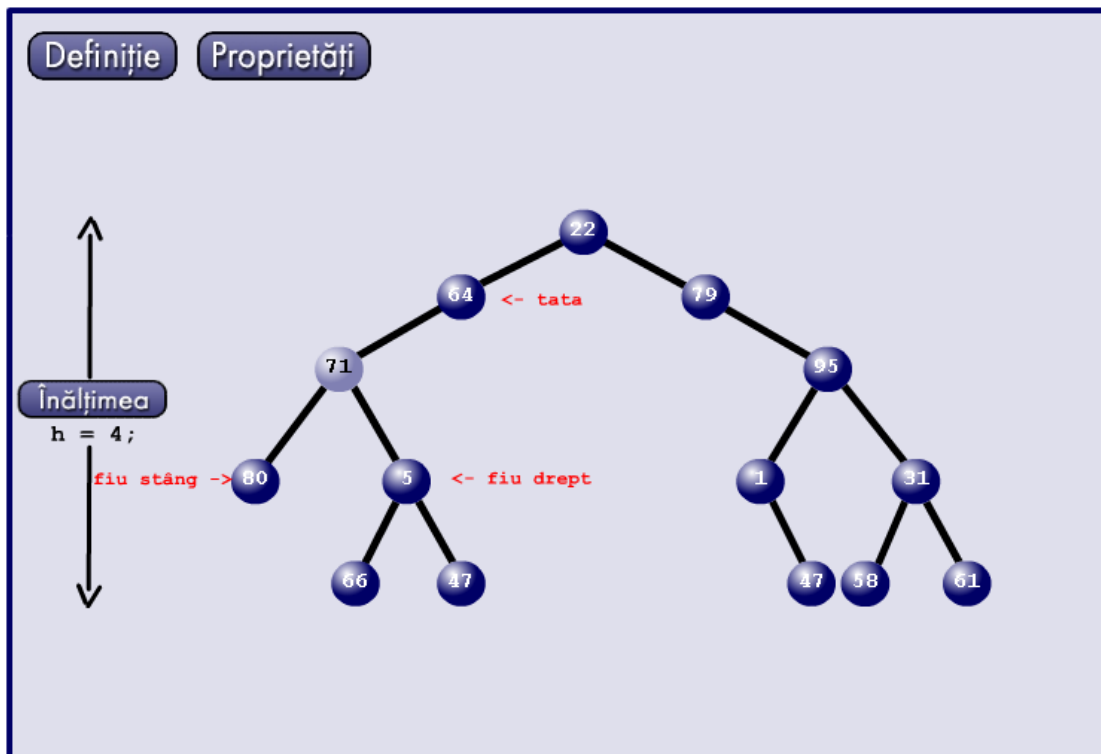
care acționat deschide o fereastră detaliu care conține definiția arborelui cu rădăcină

Nivelul

care acționat deschide o fereastră detaliu care conține definiția nivelului unui nod

După terminarea construirii arborelui, în partea stângă a acestuia, apare un nou buton, care acționat deschide fereastra detaliu corespunzătoare, precum și exemplificarea definiției pentru arborele construit

La acționarea celui de al doilea item – **Arbore binar** – apare obiectul de conținut care descrie noțiunea de arbore binar.



Și aici sunt disponibile butoanele "Definiție" și "Înălțimea" cu funcțiunile descrise mai sus. În plus mai apare și butonul

"Proprietăți" la acționarea acestui buton, într-o fereastră detaliu se descriu proprietățile specifice unui arbore binar

X Proprietăți

Proprietatea 1 :: Numărul maxim de noduri de pe nivelul i al unui arbore binar este 2^i .

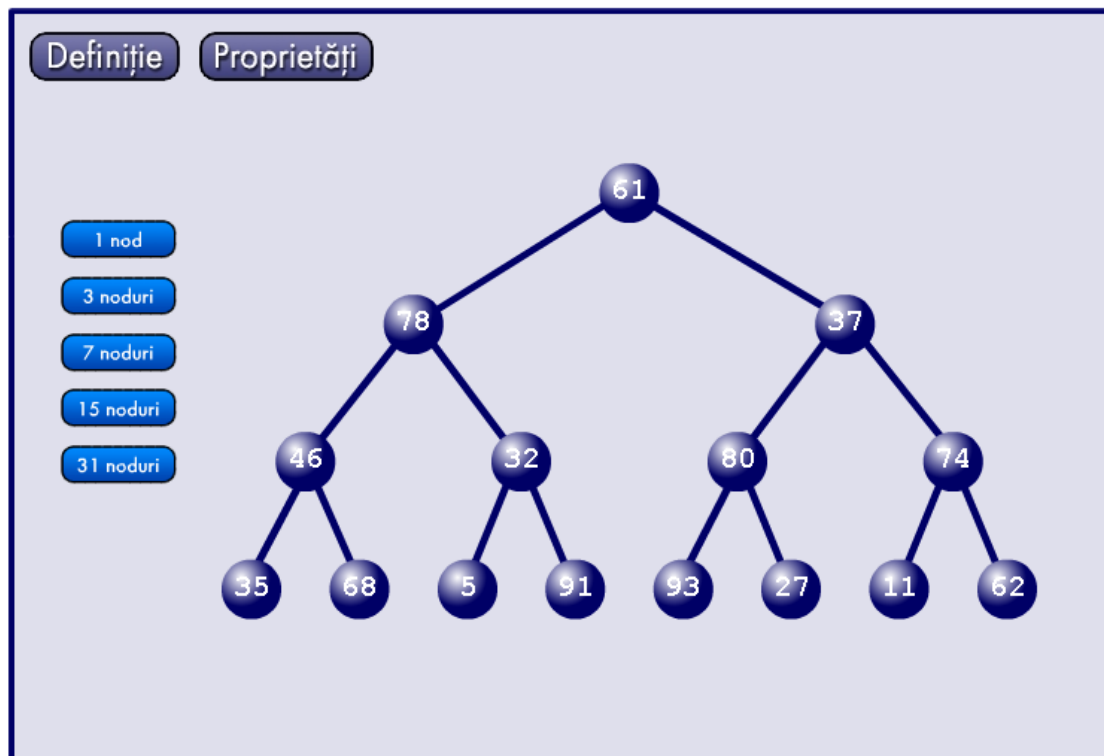
Proprietatea 2 :: Numărul maxim de noduri într-un arbore binar cu înălțimea h este $2^{h+1}-1$

Proprietatea 3 :: În orice arbore binar nevid cu n noduri noduri terminale există n-1 noduri de grad 2.

Proprietatea 4 :: Un arbore binar cu n vârfuri are înălțimea cel puțin egală cu $\lceil \log_2 n \rceil$.

Aționând asupra unui nod oarecare al arborelui sunt indicate nodurile cu care acesta este în legătura și relația dintre acestea.

La acționarea celui de al treilea item – **Arbore binar plin** – apare obiectul de conținut care descrie noțiunea de arbore binar plin. În partea stângă a ferestrei respective apar cinci butoane care indică numărul de noduri (1, 3, 7, 15, 31) pentru arborii binari plini dați ca exemplu (aici 15).

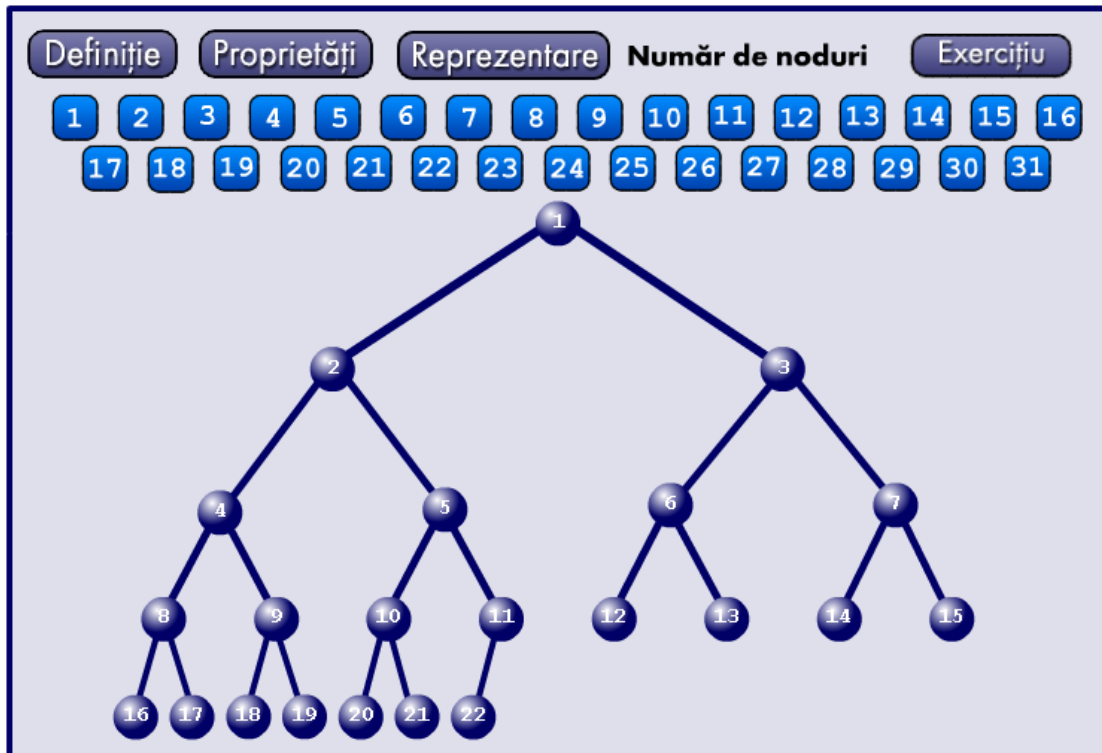


În cazul în care se acționează de două sau mai multe ori consecutiv pe același buton, vor fi generați arbori care de aceeași dimensiune, dar valorile generate în noduri vor fi altele.

Butoanele **Definiție** și **Proprietăți** îndeplinesc aceleași funcțiuni ca cele descrise mai sus.

La acționarea celui de al patrulea item – **Arbore binar complet** – apare obiectul de conținut care descrie noțiunea de arbore binar complet. În partea de sus a ferestrei apar patru butoane, două dintre ele, **Definiție** și **Proprietăți** fiind deja prezentate.

Aționarea butonului **Reprezentare** va produce apariția unei ferestre detaliu care cuprinde descrierea reprezentării unui arbore binar complet utilizând structura de date tablou unidimensional, reprezentare ce este optimă din punctul de vedere al spațiului de memorie ocupat.



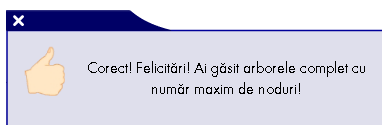
Sub aceste butoane există un număr de 31 de butoane numerotate de la 1 la 31, acționarea unuia din ele producând generarea și reprezentarea unui arbore binar complet cu numărul respectiv de noduri. Nodurile generate vor fi numerotate conform reprezentării secvențiale a unui arbore binar complet.

Acționarea butonului **Exercițiu** duce la apariția obiectului de conținut în care se testează înțelegerea noțiunii de arbore binar complet. Revenirea la fereastra anterioară se poate face în orice moment acționând butonul **Revenire** care apare în partea superioară dreapta a ferestrei.

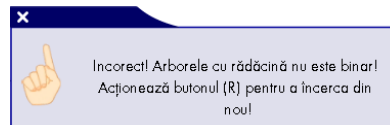
Cu ajutorul mouse-ului se selectează nodul care se dorește a fi eliminat, apoi se acționează butonul **Șterge**. Va fi eliminat atât nodul respectiv cât și întreg sub-arborele a cărui rădăcină este, ca și legătura spre nodul părinte. În permanență, în partea de jos a ferestrei sunt date informații despre numărul de noduri ce vor fi eliminate prin selecția respectivă, ca și numărul de noduri deja eliminate.

În momentul în care se consideră că exercițiul este încheiat, se acționează butonul **Gata**. Rezultatul este indicat în mod sugestiv :


Exercițiu rezolvat CORECT





Exercițiu rezolvat INCORECT





În cazul în care exercițiul nu a fost rezolvat corect, mesajul furnizat de aplicație poate să fie diferit în funcție de eroarea care a fost făcută.


Exercițiul poate fi reluat în orice moment acționând butonul .

La acționarea celui de al patrulea item – **Verificarea cunoștințelor** – reapare obiectul de conținut inițial care conține un test cu cinci exerciții recapitulative, numerotate de la 1 la 5 de forma  sub care se găsește butonul inactiv .

Exercițiile 2, 3, și 5 sunt de tip test grilă cu răspunsuri de tip “complement simplu”, adică doar o variantă de răspuns corectă. Exercițiul 1 și 4 sunt de tip interactiv, solicitând selectarea cu ajutorul mouse-ului a unui nod din reprezentarea grafică a unui graf, respectiv completarea cu valorile nodurilor unui arbore a câmpurilor din exercițiu. Exercițiile se selectează cu ajutorul mouse-ului.

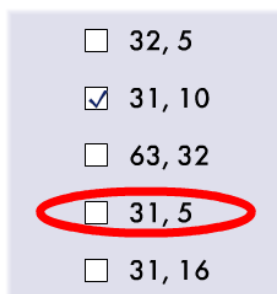
Cât timp nu s-a răspuns la toate cele cinci exerciții butonul  rămâne inactiv. Elevul poate selecta oricare din exerciții, în orice ordine, poate reveni la un exercițiu, îl poate modifica.

După rezolvarea tuturor exercițiilor butonul  devine activ, dar chiar și acum se pot face modificări în exerciții.

Finalizarea exercițiilor se face acționând butonul . Din acest moment nu se mai pot face modificări în exerciții și în dreapta butoanelor care indică numărul exercițiului este prezentat rezultatul evaluării.



La acționarea butoanelor care indică numărul exercițiului este prezentat din nou exercițiul, indicându-se, cu culoarea roșie răspunsul corect.



În același timp, în partea din dreapta jos a ferestrei sunt indicate noțiunile pe care elevul trebuie să le revadă.

3.2. MaxHeap – prezentare

În acest obiect de conținut este prezentată noțiunea de MaxHeap. Fereastra este împărțită în mai multe zone: definiția, exemple, exercițiu, definirea MaxHeap-ului ca tablou.

definiție
exemple
exercițiu

Un **MaxHeap** este un arbore binar complet în care valoarea memorată în orice nod al său este mai mare sau egală decât valorile memorate în nodurile fii ai acestuia.

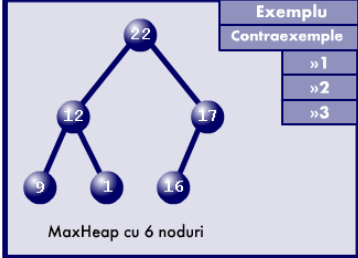
Exemplu

Contraexemplu

» 1

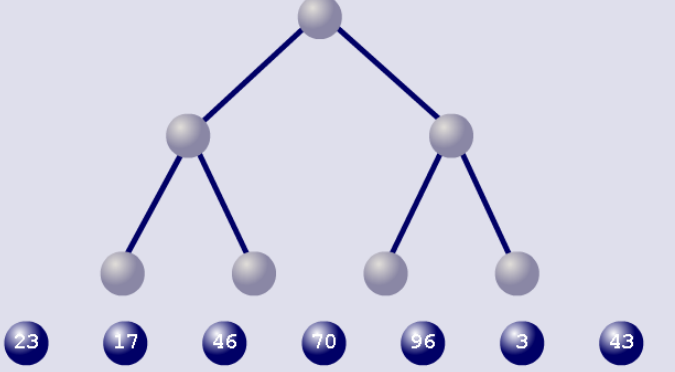
» 2

» 3



MaxHeap cu 6 noduri

Organizați valorile specificate sub forma unui MaxHeap



Utilizând reprezentarea secvențială, putem reformula definiția unui MaxHeap astfel:
 Tabloul $H[1..n]$ formează un MaxHeap dacă $H[i] \leq H[i/2]$, $i = 1, n$;

definirea ca tablou

Exemplele și contraexemplele se selectează cu ajutorul mouse-ului în orice ordine și în orice moment.

Exercițiul se rezolvă utilizând tehnica "drag_and_drop", trăgând cu mouse-ul nodurile din partea de jos a ecranului în una din locațiile arborelui. Nodurile pot fi mutate în orice ordine și locație liberă.

La completarea locațiilor apare un mesaj care indică dacă exercițiul a fost sau nu rezolvat corect.

Dacă exercițiul nu a fost rezolvat corect, se poate relua rezolvarea lui acționând butonul **R**. Și în cazul în care exercițiul a fost rezolvat corect se poate relua exercițiul, în acest caz însă exercițiul reluându-se cu alte valori ale nodurilor.

3.3. MinHeap – prezentare

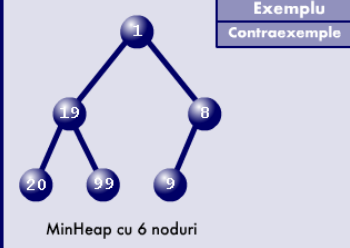
În acest obiect de conținut este prezentată noțiunea de MinHeap. Fereastra este împărțită în mai multe zone: definiția, exemple, exercițiu, definirea MinHeap-ului ca tablou.

definiție
exemple
exercițiu

Un **MinHeap** este un arbore binar complet în care valoarea memorată în orice nod al său este mai mică sau egală decât valorile memorate în nodurile fi ai acestuia.

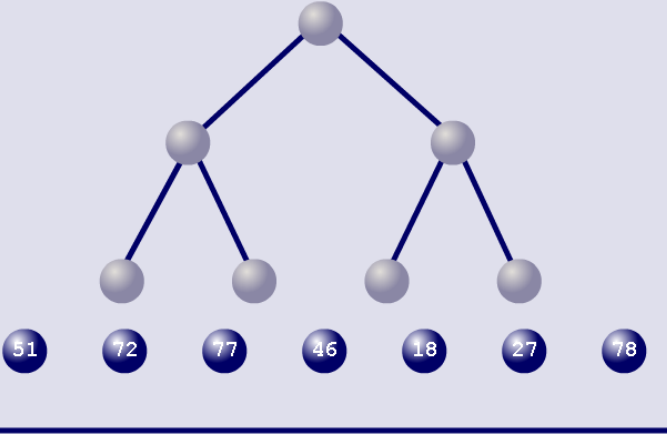
Exemplu

Contraexemplu



MinHeap cu 6 noduri

Organizați valorile specificate sub forma unui MinHeap



Utilizând reprezentarea secvențială, putem reformula definiția unui MinHeap astfel:
 Tabloul $H[1..n]$ formează un **MinHeap** dacă $H[i] \geq H[i/2]$, $i = 1, n$;

definirea ca tablou



Exemplele și contraexemplele se selectează cu ajutorul mouse-ului în orice ordine și în orice moment.

Exercițiul se rezolvă utilizând tehnica "drag_and_drop", trăgând cu mouse-ul nodurile din partea de jos a ecranului în una din locațiile arborelui. Nodurile pot fi mutate în orice ordine și locație liberă.


La completarea locațiilor apare un mesaj care indică dacă exercițiul a fost sau nu rezolvat corect.


Dacă exercițiul nu a fost rezolvat corect, se poate relua rezolvarea lui acționând butonul **R**. Și în cazul în care exercițiul a fost rezolvat corect se poate relua exercițiul, în acest caz însă exercițiul reluându-se cu alte valori ale nodurilor.


3.4. Test

Acest obiect de conținut constă într-un test cu trei exerciții recapitulative, numerotate de la 1 la 3 de forma  sub care se găsește butonul inactiv .

Cele trei exerciții sunt de tip test grilă cu răspunsuri de tip “complement simplu”, adică doar o variantă de răspuns corectă. Exercițiile se selectează cu ajutorul mouse-ului.

Cât timp nu s-a răspuns la toate cele trei exerciții butonul  rămâne inactiv. Elevul poate selecta oricare din exerciții, în orice ordine, poate reveni la un exercițiu, îl poate modifica.

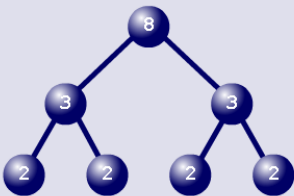
După rezolvarea tuturor exercițiilor butonul  devine activ, dar chiar și acum se pot face modificări în exerciții.

Finalizarea exercițiilor se face acționând butonul . Din acest moment nu se mai pot face modificări în exerciții și în dreapta butoanelor care indică numărul exercițiului este prezentat rezultatul evaluării.



La acționarea butoanelor care indică numărul exercițiului este prezentat din nou exercițiul, indicându-se, cu culoarea roșie răspunsul corect.

Se consideră MaxHeap-ul următor:



Care dintre următorii vectori constituie reprezentarea secvențială a acestui MaxHeap?

- H = {8, 2, 3, 3, 2, 2, 2};
- H = {8, 3, 3, 2, 2, 2, 2};
- H = {8, 3, 2, 2, 2, 2, 3};
- H = {3, 8, 3, 2, 2, 2, 2};

În același timp, în partea din dreapta jos a ferestrei sunt indicate noțiunile pe care elevul trebuie să le revadă.

3.5. Inserarea unui nod într-un heap

În acest obiect de conținut este prezentat algoritmul de inserare a unui nou nod într-un heap creat anterior. Fereastra este împărțită în mai multe zone: descrierea algoritmului în limbaj natural, animația, algoritmul în pseudocod, C++ sau Pascal, zona de comentarii. Zona de comentarii este și ea împărțită în două: comentariile care însoțesc animația respectiv variabilele care intervin în algoritm și valorile pe care acestea le iau pe parcursul desfășurării algoritmului.

descriere
algoritm
animația
algoritm

Fie H un heap cu n elemente si x valoarea ce trebuie inserată. Valoarea x va fi memorată în heap pe poziția n+1, apoi se restaurează eventual heap-ul, "promovând" valoarea x spre rădăcină, până când proprietatea de heap este restabilită.

Control animație
▶ ▶▶ ⏸ ■

```

graph TD
    18((18)) --- 15((15))
    18 --- 11((11))
    15 --- 13((13))
    15 --- 2((2))
    11 --- 9((9))
            
```

Pentru început, avem un heap ce conține 6 elemente, precum în figura de mai sus.

```

InsertHeap (int &n, int x)
{
    int tata, fiu, aux;
    n<-n+1;
    fiu <- n; tata <- n/2;
    H[fiu] <- x;
    cat timp (tata>0 si H[tata]<H[fiu])
    {
        aux <- H[fiu];
        H[fiu] <- H[tata];
        H[tata] <- aux;
        fiu <- tata; tata <- fiu/2;
    }
}
            
```

pseudocod C++ pascal

n = 6;
x = nedefinit;
fiu = nedefinit;
tata = nedefinit;
H = {18, 15, 11, 13, 2, 9};

comentarii

Complexitate

Animația poate fi controlată în sensul executării acesteia în mod continuu sau în modul “pas cu pas” pentru a putea fi urmărită fiecare etapă a desfășurării algoritmului de inserare a unui nou nod într-un heap. Pe parcursul desfășurării animației, o bară portocalie indică în fiecare moment instrucțiunea corespunzătoare din pseudocod (C++ sau Pascal). Valorile numerice aflate pe nodurile heap-ului sunt de două tipuri: cele care sunt deja pe pozițiile lor (indicate prin font normal) și cele care încă nu au ajuns în heap pe pozițiile finale (**bold**).

În același timp zona de comentarii se modifică, comentând ceea ce se întâmplă și arătând valorile curente ale variabilelor.

În partea din dreapta jos a ferestrei butonul Complexitate permite afișarea unei ferestre detaliu în care se descrie modul de calcul al complexității algoritmului.

În orice moment este disponibil butonul Obiective la acționarea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.

3.6. Crearea unui heap prin inserări repetate

În acest obiect de conținut este prezentat algoritmul de creare a unui heap pornind de la heap-ul vid, prin inserări repetate de noduri conform algoritmului descris în obiectul de conținut 3.5. Fereastra este împărțită în trei zone: algoritmul în limbaj natural, animația, algoritmul în pseudocod (C++, Pascal).

descriere
algoritm

animația

algoritm

O primă soluție este de a insera succesiv elementele în heap plecând inițial de la heap-ul vid.

```

CreareHeap()
{
  int i;
  pentru i de la 1 la n
    InsertHeap(i,H[i]);
}
        
```

n = 7 ;

Introduceți valoarea nodului 5 :

x =

Inițial, în fereastra unde se va desfășura animația există doar solicitarea de a introduce valoarea lui n, numărul de noduri care vor fi inserate în heap, indicându-se și valorile corecte care pot fi introduse.

n = ;

1 <= n <= 31

În cazul în care se introduce o valoare eronată, la acțiunea butonului sau a tastei ENTER apare un mesaj și se solicită introducerea unei noi valori.

După introducerea unei valori corecte în fereastră apare schema unui arbore complet cu n noduri, iar în partea de jos a ferestrei sunt solicitate valorile nodurilor. La introducerea unei valori corecte, heap-ul se reface automat și se solicită introducerea nodului următor. După introducerea tuturor nodurilor un mesaj indică crearea cu succes a heap-ului și apare butonul care permite reluarea operației de creare a unui heap.

În partea din dreapta jos a ferestrei butonul permite afișarea unei ferestre detaliu în care se descrie modul de calcul al complexității algoritmului.

În orice moment este disponibil butonul la acțiunea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.

3.7. Combinarea a două heap-uri

În acest obiect de conținut este prezentat algoritmul de combinare a unui nou nod cu două heap-uri create anterior, astfel încât să rezulte tot un heap. Fereastra este împărțită în mai multe zone: descrierea algoritmului în limbaj natural, animația, algoritmul în pseudocod, C++ sau Pascal, zona de comentarii.

descriere
algoritm

animația

algoritm

O astfel de operație creează la fiecare pas i heap-ul cu rădăcina $H[i]$, combinând două heap-uri de dimensiuni apropiate, heap-ul cu rădăcina $2 \cdot i$ cu heap-ul cu rădăcina $2 \cdot i + 1$ și cu elementul $H[i]$. Nodul tată (inițial i) este "retrogradat" fiind înlocuit cu fiul cu valoarea cea mai mare până când proprietatea de heap va fi restabilită.

Control animație

▶
▶▶
⏸
■

Oprește animația

3

15
12 14

19
9 1

Vom ilustra un exemplu de combinare a două heap-uri cu nodul i , care în cazul acesta are cheia 3.

```

CombHeap (int i, int n)
{
  int fiu, tata, aux;
  tata <- i; fiu <- 2*i;
  cat timp fiu<=n
  {
    daca fiu+1<=n atunci
      daca H[fiu] < H[fiu+1] fiu <-fiu+1;
      daca H[tata] < H[fiu] atunci
        {
          aux <- H[tata];
          H[tata] <- H[fiu];
          H[fiu] <- aux;
          tata <- fiu; fiu <- fiu*2;
        }
      altfel fiu <- n+1;
    }
  }

```

pseudocod
C++
pascal

Complexitate

comentarii

Animația poate fi controlată în sensul executării acesteia în mod continuu sau în modul "pas cu pas" pentru a putea fi urmărită fiecare etapă a desfășurării algoritmului de inserare a unui nou nod într-un heap. Pe parcursul desfășurării animației, o bară portocalie indică în fiecare moment instrucțiunea corespunzătoare din pseudocod (C++ sau Pascal). Valorile numerice aflate pe nodurile heap-ului sunt de două tipuri: cele care sunt deja pe pozițiile lor (indicate prin font normal) și cele care încă nu au ajuns în heap pe pozițiile finale (**bold**).

În același timp zona de comentarii se modifică, comentând ceea ce se întâmplă.

În partea din dreapta jos a ferestrei butonul Complexitate permite afișarea unei ferestre detaliu în care se descrie modul de calcul al complexității algoritmului.

În orice moment este disponibil butonul Obiective la acționarea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.

3.8. Crearea unui heap prin combinare

În acest obiect de conținut este prezentat algoritmul de creare a unui heap prin combinări repetate de heap-uri conform algoritmului descris în obiectul de conținut 3.7. Fereastra este împărțită în trei zone: algoritmul în limbaj natural, animația, algoritmul în pseudocod (C++, Pascal). Inițial, în fereastra unde se va desfășura animația există doar solicitarea de a introduce valoarea lui n, numărul de noduri, indicându-se și valorile corecte care pot fi introduse.

descriere algoritm

animația

algoritm

O strategie eficientă de construcție a unui heap se bazează pe ideea de echilibrare. Apelul procedurii CombHeap(i,n) poate fi interpretat ca o combinare de două heap-uri: un heap cu n elemente și un heap format numai din elementul x. Putem construi heap-ul cu rădăcina H[i] combinând la fiecare pas două heap-uri cu dimensiuni apropiate, heap-ul cu rădăcina 2i, heap-ul cu rădăcina 2i+1 și cu elementul H[i];

```

CreareHeap()
{
  int i;
  pentru i de la n/2 la 1
    CombHeap(i, n);
}
                    
```

pseudocod
C++
pascal

n = 7 ;
Pasul următor OK

Complexitate

În cazul în care se introduce o valoare eronată, la acționarea butonului OK sau a tastei ENTER apare un mesaj și se solicită introducerea unei noi valori.

După introducerea unei valori corecte în fereastră apare schema unui arbore complet cu n noduri, iar imediat sub valoarea lui n începe dialogul prin care sunt solicitate valorile nodurilor. La introducerea unei valori corecte, aceasta este introdusă în arbore pe poziția imediat următoare, conform notației secvențiale și se solicită introducerea valorii următoare. După introducerea tuturor valorilor, pentru a trece la pasul următor – combinarea heap-urilor – se solicită acționarea butonului OK. La fiecare acționare a butonului heap-ul se reface pas cu pas. În momentul în care heap-ul a fost complet restaurat apare butonul R care permite reluarea operației de creare a unui heap prin combinări repetate.

În partea din dreapta jos a ferestrei butonul Complexitate permite afișarea unei ferestre detaliu în care se descrie modul de calcul al complexității algoritmului.

În orice moment este disponibil butonul Obiective la acționarea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.

3.9. Extragerea unui nod dintr-un heap

În acest obiect de conținut este prezentat algoritmul de extragere a unui nod dintr-un heap, urmată de restabilirea proprietății de heap pentru nodurile rămase. Fereastra este împărțită în mai multe zone: descrierea algoritmului în limbaj natural, animația, algoritmul în pseudocod, C++ sau Pascal, zona de comentarii.

descriere
algoritm

animația

algoritm

Fie H un heap cu n elemente. Elementul ce va fi extras ocupă poziția 1 în heap. În locul lui va fi amplasat elementul de pe poziția n. Noul element aflat pe poziția 1 va fi "retrogradat" până când proprietatea de heap va fi restabilită.

Control animație

▶
▶▶
⏸
■

Următorul pas

Elementul de pe ultima poziție (în acest caz 5) va înlocui pe cel ce trebuie extras.

```

ExtractHeap(int &n)
{
  int aux, tata, fiu;
  tata <- 1; fiu <- 2;
  H[1] <- H[n];
  n <- n-1;
  cat timp fiu<=n
  {
    daca fiu+1<=n atunci
      daca H[fiu]<H[fiu+1] atunci fiu<-fiu+1;
      daca H[tata]<H[fiu] atunci
        {
          aux <- H[tata];
          H[tata] <- H[fiu];
          H[fiu] <- aux;
          tata <- fiu; fiu <- fiu*2;
        }
      else fiu = n+1;
    }
  }

```

pseudocod
C++
pascal

Complexitate

comentarii

Animația poate fi controlată în sensul executării acesteia în mod continuu sau în modul "pas cu pas" pentru a putea fi urmărită fiecare etapă a desfășurării algoritmului de extragere a unui nod dintr-un heap și refacerea proprietății de heap. Pe parcursul desfășurării animației, o bară portocalie indică în fiecare moment instrucțiunea corespunzătoare din pseudocod (C++ sau Pascal). În același timp zona de comentarii se modifică, comentând ceea ce se întâmplă.

Animația se poate relua acționând butoanele de control a animației ▶ sau ▶▶.

În partea din dreapta jos a ferestrei butonul Complexitate permite afișarea unei ferestre detaliu în care se descrie modul de calcul al complexității algoritmului.

În orice moment este disponibil butonul Obiective la acționarea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.

3.10. Coada cu prioritate

În acest obiect de conținut este prezentată coada cu prioritate prin intermediul unei aplicații obișnuite într-un laborator de informatică și anume utilizarea unei imprimante de rețea. Fereastra este împărțită în mai multe zone: definiția unei cozi cu prioritate, animația, zona de comentarii.

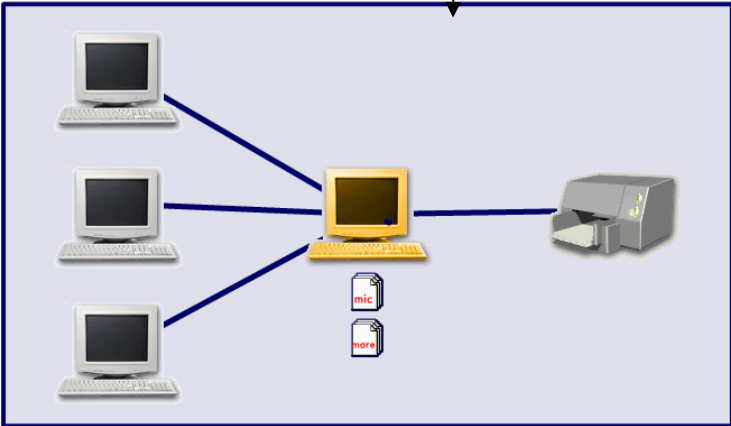
definiție
animația
comentarii

Definiție

O coadă cu prioritate este o structură de date abstractă formată din elemente care au asociată o valoare numită cheie sau prioritate și care suportă următoarele operații:

- **Insert(Q,x)**: inserează elementul x în coada cu prioritate Q;
- **ExtractMax(Q)**: extrage elementul de valoare maximă din coada cu prioritate Q.

Din coadă se alege fișierul care are prioritate maximă. Acesta este trimis spre a fi listat. Ca urmare a comenzilor de tipărire, alte fișiere se adaugă în coadă spre a fi listate.



Control animație

▶▶▶ || ■

Următorul pas

Complexitate



Animația poate fi controlată în sensul executării acesteia în mod continuu sau în modul “pas cu pas” pentru a putea fi urmărită fiecare etapă a modului de lucru în care se desfășoară selectarea documentelor trimise spre imprimantă atunci când aceste documente au asociată o anumită prioritate. Pe parcursul desfășurării animației în zona de comentarii sunt afișate comentarii asupra desfășurării evenimentelor.

Animația se poate relua acționând butoanele de control a animației ▶ sau ▶▶.


În partea din dreapta jos a ferestrei butonul Complexitate permite afișarea unei ferestre detaliu în care sunt comparate diferite metode de implementare a unei cozi cu prioritate (vector, vector ordonat, listă simplu înlănțuită, heap).


În orice moment este disponibil butonul Obiective la acționarea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.


3.11. Test recapitulativ coada cu prioritate

Acest obiect de conținut constă într-un test cu trei exerciții recapitulative, numerotate de la 1 la 3 de forma  sub care se găsește butonul inactiv .

Cele trei exerciții sunt de tip test grilă cu răspunsuri de tip “complement simplu”, adică doar o variantă de răspuns corectă. Exercițiile se selectează cu ajutorul mouse-ului.

Cât timp nu s-a răspuns la toate cele trei exerciții butonul  rămâne inactiv. Elevul poate selecta oricare din exerciții, în orice ordine, poate reveni la un exercițiu, îl poate modifica.

După rezolvarea tuturor exercițiilor butonul  devine activ, dar chiar și acum se pot face modificări în exerciții.

Finalizarea exercițiilor se face acționând butonul . Din acest moment nu se mai pot face modificări în exerciții și în dreapta butoanelor care indică numărul exercițiului este prezentat rezultatul evaluării.



La acționarea butoanelor care indică numărul exercițiului este prezentat din nou exercițiul, indicându-se, cu culoarea roșie, răspunsul corect.



Care este cea mai eficientă modalitate de implementare a unei cozi cu prioritate, din punct de vedere a complexității ?


- listă simplu înlănțuită ordonată
- heap
- listă dublu înlănțuită ordonată
- vector ordonat

3.12. Heapsort

În acest obiect de conținut este prezentat algoritmul de sortare a unui vector utilizând structura de date heap și noțiunile învățate în obiectele de conținut 3.2, 3.7, 3.8, 3.9. Fereastra este împărțită în mai multe zone: enunțul problemei, animația, algoritmul în pseudocod, C++ sau Pascal, zona de comentarii.



Animația poate fi controlată în sensul executării acesteia în mod continuu sau în modul “pas cu pas” pentru a putea fi urmărită fiecare etapă a desfășurării algoritmului heapsort. Pe parcursul desfășurării animației, o bară portocalie indică în fiecare moment etapa corespunzătoare din pseudocod (C++ sau Pascal). În același timp zona de comentarii se modifică, comentând ceea ce se întâmplă.

Animația se poate relua acționând butoanele de control a animației  sau .


Butonul  permite afișarea unei ferestre detaliu în care se face o comparație între șapte metode de sortare, cele studiate în liceu.


În orice moment este disponibil butonul  la acționarea căruia apare fereastra detaliu ce descrie obiectivele urmărite în acest obiect de conținut.


3.13. Test recapitulativ Heapsort

Acest obiect de conținut constă într-un test cu trei exerciții recapitulative, numerotate de la 1 la 3 de forma  sub care se găsește butonul inactiv .

Cele trei exerciții sunt de tip test grilă cu răspunsuri de tip “complement simplu”, adică doar o variantă de răspuns corectă. Exercițiile se selectează cu ajutorul mouse-ului.

Cât timp nu s-a răspuns la toate cele trei exerciții butonul  rămâne inactiv. Elevul poate selecta oricare din exerciții, în orice ordine, poate reveni la un exercițiu, îl poate modifica.

După rezolvarea tuturor exercițiilor butonul  devine activ, dar chiar și acum se pot face modificări în exerciții.

Finalizarea exercițiilor se face acționând butonul . Din acest moment nu se mai pot face modificări în exerciții și în dreapta butoanelor care indică numărul exercițiului este prezentat rezultatul evaluării.



La acționarea butoanelor care indică numărul exercițiului este prezentat din nou exercițiul, indicându-se, cu culoarea roșie, răspunsul corect.

Se consideră următorul vector cu 6 elemente organizate ca un MaxHeap:

$$H = \{22, 11, 8, 5, 10, 6\};$$

Pentru a sorta vectorul cu ajutorul heap-urilor extragem la fiecare pas elementul de valoare maximă și restaurăm heap-ul.

Care dintre următoarele variantele ilustrează heap-ul obținut după executarea unui pas ?

- H = {6, 11, 8, 5, 10};
- H = {11, 10, 8, 5, 6};
- H = {11, 6, 8, 5, 10};
- H = {11, 10, 8, 6, 5};

4. Bibliografie

- **Mateescu (Cercez) E., Maxim I.**, Arbori, Editura Țara Fagilor, Suceava, 1996
- **Horowitz E., Sahni S., Anderson-Freed S.**, Fundamentals of Data Structures in C, Computer Science Press, New York, 1993
- **Cormen Th., Leiserson Ch., Rivest R.**, Introduction to Algorithms, MIT, 1990
- **Cercez Em.**, Informatica, Culegere de probleme pentru liceu, Editura Polirom, Iași, 2002