Teoria PAA text

Se dorește se codifică fie caracter printr-o secvență de cifre binare zero și unu astfel încât codul în caracter să nu fie prefix pentru codul nici unui alt caracter .

1. Prefix

Căutarea unei key oarecare într-un arbore AVL este în cel mai rău caz de complexitatea O(log 2N), pe când într-un arbore binar ordonat ne echilibrat este în cel mai rău caz de O(N).

1. True

Care din următoarele argumente este esențial pentru alege un arbore B în loc de un arbore AVL pentru indexare unor baze de date .

1. **Datele se preiau din memorie externă în blocuri**
2. Înregistrările în bazele de date sunt deja sortate după o cheie primar
3. Bazele de date un număr mare de înregistrări
4. Arbori B necesită mai puțină memorie decât arbori AVL

Un arbore bă crește înălțime doar atunci când:

1. Se adaugă un nou nod într-o pagină terminală
2. În niciuna din celelalte variante
3. **Are loc o scindare a paginii rădăcină**
4. Se adaugă nouă pagină terminal

Un cod prefix (care are proprietatea de prefix) poate fi reprezentat printr-un arbore binar dacă nodurilor "terminale" ale arborelui li se asociază caracterele originale ale alfabetului iar ramurilor

1. Ne terminale
2. Cu key pare
3. **Terminale**
4. Cu ponderea cea mai mare

Numărul de biți necesar codarii cuvântul următor, folosind codul Hoffmann este: ABRACADABRA

Dacă presupunem că toate caracterele din cuvântul "ABRACADABRA" au frecvențe egale, putem construi un arbore Huffman echilibrat. În acest caz, arborele Huffman va avea toate ramurile la aceeași adâncime și codurile pentru fiecare caracter vor avea aceeași lungime.
Pentru cuvântul "ABRACADABRA" cu frecvențe egale, putem utiliza un sistem binar pentru codificare, în care fiecare caracter este reprezentat printr-un cod de lungime fixă, cum ar fi 4 biți. Astfel, fiecare caracter va fi codificat în 4 biți, iar numărul total de biți pentru codarea întregului cuvânt va fi de 4 biți × numărul de caractere din cuvânt.
În cazul cuvântului "ABRACADABRA" (cu 11 caractere), numărul total de biți necesar pentru codare, în acest caz, ar fi de 4 biți × 11 caractere = 44 biți.
Este important de menționat că aceasta este o situație simplificată și

presupune că frecvențele caracterelor sunt egale, ceea ce este neobișnuit în majoritatea cazurilor reale. În practică, în codurile Huffman, caracterelor cu frecvențe mai mari li se atribuie coduri mai scurte, ceea ce duce la o eficiență mai mare a codificării.

Un arbore B își reduce dimensiunea doar dacă:

1. **Să ștergi un loc dintr-o pagină terminală**
2. Se face o echilibrare
3. Rădăcina este redusă la dimensiunea zero

Într-un arbore optim, nodurile la care accesul se face mai frecvent devin noduri cu pondere mai "mare", cele vizitate mai rar, noduri cu pondere mai "mica".

Arbori B sunt adesea utilizați pentru:

1. **Gestionarea unor date aflate în memorie externă**
2. Gestionarea unor date fără a le reține într-un mod ordonat după key
3. Gestionarea unui număr redus de date
4. Implementarea dicționarelor de cuvinte

Care din următoarele afirmații este adevărată:

1. Cu cat este mai mare ordinul N al unui arbore B, cu atât numărul scindării lor de pagini este mai mare
2. **Cu cat este mai mic cu Ordinul N al unui arbore B, cu atât**

numărul schimbărilor de pagini este mai mic
3. Numărul scindării lor de pagini este independentă Ordinul N al arborelui B
4. Cu cat este mai mare ordinul N al unui arbore B cu atât numărul și dărilor de pagini este mai mic

Lungimile drumurilor de la rădăcină la oricare nod terminal sunt egale într-un arbore B.
True ??

Se numește arborele binar optim, arborele a cărui structură conduce
un cost minim

Care din următoarele afirmații despre codurile Huffman este adevărată:
1. **Algoritmii de tip Huffman static au dezavantajul că necesită cunoașterea prealabilă a frecvențelor de apariție pentru fiecare simbol din sursă**
2. Codarea Huffman poate prezenta pierderi de informații în anumite cazuri
3. **Codarea Huffman este folosită pentru compresie**
4. **In codarea Huffman niciun cod nu este prefixul altui cod**

Ordinul unui grafic este:
1. Numărul de arce pe care acesta le conține
2. **Numărul de noduri pe care acesta le conține**
3. Egal cu gradul Grafului
4. Numărul maxim de arce

incidente unui anumit nod din gra

Care este numărul de arce dintr-un graf complet cu N noduri?

1. N^N
2. N-1
3. N*N/2
4. **(N*(N-1))/2**

Care din următoarele structuri este folosită eficient pentru traversarea prin cuprindere a unui graf.

1. Coadă cu priorități
2. Stiva
3. Arborele
4. **Coada**

Un graf conex a ciclic este un arbore.
True.

Un graf conex aciclic este echivalent cu un arbore. Un arbore este un graf conex aciclic, adică nu conține cicluri și fiecare pereche de noduri este conectată printr-un singur drum unic.

Prin urmare, afirmația "Un graf conex aciclic este un arbore" este adevărată.

Care este numărul maxim de arce într-un graf neorientat aciclic cu N noduri:
N
2N-1
N-1
N+1

Un arbore este un graf neorientat aciclic care conține N noduri și N-1 muchii. Aceasta înseamnă că, într-un

arbore, fiecare nod, cu excepția rădăcinii, este conectat la exact o muchie și există o singură cale între oricare două noduri.

Într-un arbore AVL toate nodurile frunze se găsesc la aceeași adâncime.
False, adancimea poate sa difere cu maxim 1 nivel

Într-un arbore B toate paginile terminale apar la același nivel .
True

Într-un arbore B, toate paginile terminale (sau frunzele) apar la același nivel. Un arbore B este un arbore echilibrat în care toate paginile terminale se află la aceeași adâncime. Aceasta este una dintre proprietățile esențiale ale unui arbore B, care asigură că toate înregistrările din baza de date (reprezentate de paginile terminale) sunt distribuite uniform și eficient în structura arborelui.

Arborele de acoperire minim este întotdeauna unic pentru un graf dat.
False

Arborele de acoperire minim (MST - Minimum Spanning Tree) nu este întotdeauna unic pentru un graf dat. Un graf poate avea mai multe arbori de acoperire minim, în funcție de algoritmul utilizat pentru a-l construi și de posibile egalități de ponderi între muchii.

Există mai multe algoritme pentru construirea unui MST, cum ar fi algoritmul lui Prim și algoritmul lui Kruskal. Aceste algoritme pot conduce la arbori de acoperire minim diferiți pentru același graf, în funcție de ordinea de explorare a muchiilor și alegerea muchiilor cu ponderi minime.

Pentru traversarea în in ordine nodurile se prelucrează:
Prima dată când sunt întâlnite
Ultima dată când sunt întâlnite
Se prelucrează un nou terminal la prima întâlnire și unul interior la doua întâlnire
A doua oară când sunt întâlnite

Traversarea unui grav poate fi efectuată fără reține nodurile vizitate.
False.

Traversarea unui graf, în general, nu poate fi efectuată fără reținerea nodurilor vizitate. Pentru a evita ciclurile și pentru a asigura că fiecare nod este vizitat o singură dată, este necesară menținerea unei evidențe a nodurilor deja vizitate

Care din următoarele structuri este folosită eficient pentru traversarea prin cuprindere a unui graf ?

1. Arborele
2. Stiva
3. Coada cu priorități
4. **Coada**

Numărul de biți necesar coduri cuvântului următor folosind codul

Huffman este: INDISTINCTIBILI (40 ? 42 ?)

În cazul unui arbore B inserția unui nod într-o pagină plină duce la:

1. Echilibrare a arborelui
2. Contopirea pagini cu pagina părinte
3. **Scindarea pagini**

În cazul unui arbore B, inserția unui nod într-o pagină plină duce la scindarea paginii. Acest proces implică împărțirea paginii în două pagini separate, iar cheia medie este promovată la pagina părinte pentru a menține proprietățile arborelui B.

Pentru traversarea în pre ordin nodurile se prelucrează:

1. A doua oară când sunt întâlnite
2. **Prima dată când sunt întâlnite**
3. Se prelucrează un nod terminal la prima întâlnire și unul interior la adoua întâlnire
4. Ultima dată când sunt întâlnite

Arbori B sunt adesea utilizați pentru:

1. **Gestionarea unor date aflate în memorie externă**
2. Gestionarea unor date fără a le reține într-un mod ordonat după chei
3. Implementarea dicționarelor de cuvinte
4. Gestionarea unui număr redus de date

Căutarea unei chei oarecare într-un arbore AVL este cel mai rău caz de complexitate O(log 2 de N), pe când

într-un arbore binar ordonat neechilibrat este în cel mai rău caz de O(N).
True

Un arbore B crește înălțime doar atunci când:

1. Se adaugă un nou nod într-o pagină terminală
2. În nici una din celelalte variante
3. **Are loc o scindare a paginii rădăcină**
4. Se adaugă nouă pagină terminală

În cazul unui arbore B, inserția unui nou dintr-o pagină plină duce la scindarea paginii.
True

Care din următoarele afirmații este adevărată:

1. Cu cat este mai mare cu Ordinul N al unui arbore B, cu atât numărul scindarilor de pagini este mai mare
2. Cu cat este mai mic ordinul al unui arbore B, cu atât numărul scindării lor de pagini este mai mic
3. Numărul scindării lor de pagini este independent de ordinul al arborelui B
4. **Cu cat este mai mare ordinul N al unui arbore B, cu atât numărul scindării lor de pagini este mai mic.**

Arborele B este un arbore echilibrat din punct de vedere al înălțimii.

True

În cazul unui arbore B, toate cheile terminale se află într-adevăr pe același nivel. Prin urmare, afirmația că "Arborele B este un arbore echilibrat din punct de vedere al înălțimii" este de fapt adevărată.

Un arbore B isi reduce dimensiunea doar dacă:
1. Se face echilibrare
2. Rădăcina este redusă la dimensiune zero
3. Să șterge un nod dintr-o pagină terminală
???????

Un arbore B crește înălțime doar atunci când:

1. Se adaugă un nou Nord într-o pagină terminală
2. Se adaugă o nouă pagină terminală
3. **Are loc o scindarea pagini rădăcină**
4. În niciuna din celelalte variante

Arbori AVL Se comportă la fel ca arborii binari ordonați simpli mai puțin în cazul operațiilor de:

1. **Inserții și suprimare de chei**
2. Calcul al înălțimii
3. Inițializare

Într-un arbore B, de ordinul N, o pagină poate avea maxim

1. **2N-1 chei**
2. 2N chei

3. N chei
4. (N-1)/2 chei

Câte ar cere un graf neorientat complet cu 12 noduri ?
Un graf neorientat complet cu N noduri are N * (N - 1) / 2 arce. În cazul nostru, cu 12 noduri, numărul de arce este:
12 * (12 - 1) / 2 = 66

Un pointeri este:

1. **O variabilă care stochează adresa altei variabile**
2. O variabilă care stochează adresa unei instrucțiuni
3. Un cuvânt cheie folosit pentru a declara o variabilă

Care este numărul de arce dintr-un graf conex, aciclic cu 12 noduri?
11
Dacă un graf este conex și aciclic, este un arbore. Un arbore cu N noduri are exact N-1 arce. Prin urmare, în cazul nostru, un graf conex, aciclic cu 12 noduri va avea 12 - 1 = 11 arce. Deci, numărul de arce într-un graf conex, aciclic cu 12 noduri este 11.

Care este numărul minim de arce care pot fi adăugate pentru a transforma un graf cu 11 componente conexe într-un graf conex.
Pentru a transforma un graf cu 11 componente conexe într-un graf conex, trebuie să adăugăm suficiente arce astfel încât fiecare componentă conexă să fie conectată cu cel puțin una dintre celelalte componente. Având în vedere că avem 11

componente conexe, trebuie să adăugăm cel puțin 10 arce pentru a conecta fiecare componentă cu celelalte.
Deci, numărul minim de arce care pot fi adăugate pentru a transforma un graf cu 11 componente conexe într-un graf conex este 10.

Fie X un nod într-un arbore binar, X are doi copii . Fie nodul terminal Y succesorul lui X în parcurgerea în inordine . Care din următoarele este adevărată?

1. Y nu are copil în dreapta
2. Y este copilul din stânga a lui X
3. Nici un răspuns nu este corect
4. **Y este copilul din dreapta lui X**

Într-un arbore binar în parcurgerea în in ordine, nodurile sunt prelucrate în ordinea stânga-rădăcină-dreapta. Dacă nodul X are doi copii, înseamnă că copilul din stânga a lui X este prelucrat înaintea lui X, iar copilul din dreapta a lui X este prelucrat după X. Succesorul lui X în parcurgerea în in ordine este primul nod care este prelucrat după X, deci este copilul din dreapta a lui X. Prin urmare, afirmația corectă este că Y este copilul din dreapta a lui X.

Care este numărul maxim de argint un graf ne orientat, la ciclic cu 12 noduri ?
Numărul maxim de arce într-un graf neorientat, aciclic cu N noduri este dat de formula (N * (N - 1)) / 2. Aplicând această formulă pentru N = 12, obținem:

(12 * (12 - 1)) / 2 = (12 * 11) / 2 = 132 / 2 = 66

Numărul de biți necesar coduri cuvântului următor folosind coduri Hoffmann este:
AGRAMATICALITATEA - 66?

malloc(4); va aloca un spatiu de memorie de 4 octeti (B) si il va initializa cu 0?
False.
Funcția `malloc(4)` va aloca un spațiu de memorie de 4 octeți, dar nu va inițializa acest spațiu cu zero. Funcția `malloc` din limbajul C nu garantează o inițializare specifică a memoriei alocate. Conținutul memoriei alocate de `malloc` este necunoscut și poate conține valori nedefinite sau rămășițe de date anterioare. Dacă doriți să inițializați memoria alocată cu zero, puteți utiliza funcția `calloc`.

Numărul maxim de noduri dintr-un arbore binar de înălțime h este?

1. h^2
2. 2^h-1
3. (h+1)^2
4. **2^(h-1)**
5. 2^(h+1)-1

nivelul 1 avem radacina max 1 -> 2^(1-1)
nivelul 2 maxim 2 fii -> 2^(2-1)
la nicelul 3 maxim 4 fii -> 2^(3-1)
rezulta 2^(h-1)

Un minut oarecare al unui arbore binar are:

1. Oricati fii

2. Exact doi fii
3. **Cel mult 2 fii**
4. Cel putin 2 fii

Câte arce are un graf n orientat complet cu 4 noduri:
Un graf neorientat complet este un graf în care fiecare nod este conectat prin arce cu toate celelalte noduri din graf. Având 4 noduri, fiecare nod trebuie să fie conectat cu celelalte 3 noduri (deoarece nu se permit bucle sau arce multiple între aceleași perechi de noduri). Astfel, numărul total de arce este 4 * (4-1) / 2 = 6.

Care din următoarele variante reprezintă o declarării corectă unui pointer in C ?
1. Ptr x
2. Int &x
3. **Int *x**
4. Int x

Lungimile drumurilor de la rădăcină la un nod terminal sunt egale într-un arbore B.
True

Care funcții este adecvată pentru a dealoca memorie limbajul C ?
1. clear
2. **Free**
3. Delete
4. Remove

Care este numărul maxim de arce într-un vreau să ne orientat a ciclic cu 5 noduri ?
Într-un graf neorientat aciclic cu N

noduri, numărul maxim de arce este dat de formula: (N * (N - 1)) / 2.

În cazul unui graf neorientat aciclic cu 5 noduri, numărul maxim de arce este: (5 * (5 - 1)) / 2 = (5 * 4) / 2 = 20 / 2 = 10.

Deci, numărul maxim de arce într-un astfel de graf este 10.

Care este diferența maximă de nivel între nodurile frunză ale unui arbore AVL

1. 0
2. n, unde N este numărul de noduri
3. log(n), unde N este numărul de noduri
4. **1**

Care este numărul maxim posibil de arce într-un graf neorientat, bipartit cu 13 moduri.

Pentru un graf bipartit cu m noduri în prima parte și n noduri în a doua parte, numărul maxim de arce este m * n. În cazul unui graf bipartit cu 13 noduri, presupunând că are x noduri în prima parte și 13 - x noduri în a doua parte, numărul maxim de arce va fi x * (13 - x).

Pentru a determina numărul maxim posibil de arce, trebuie să găsim valoarea lui x care maximizează expresia x * (13 - x). Acest lucru se întâmplă atunci când x este egal cu jumătatea lui 13, adică 6.5. Deoarece numărul de noduri trebuie să fie întreg,

vom folosi atât valoarea 6, cât și 7 pentru x.

Pentru x = 6, numărul maxim de arce este 6 * (13 - 6) = 42.
Pentru x = 7, numărul maxim de arce este 7 * (13 - 7) = 42.

Deci, numărul maxim posibil de arce într-un graf neorientat, bipartit cu 13 noduri este 42.

Câte arce are un graf neorientat, complet cu 4 noduri? (6)
Un graf neorientat complet cu N noduri are (N * (N - 1)) / 2 arce. În cazul unui graf neorientat complet cu 4 noduri, numărul de arce este:
(4 * (4 - 1)) / 2 = 6

Deci, un graf neorientat complet cu 4 noduri are 6 arce.

Numărul de biți necesar cu Dory cuvântului următor folosind coduri Hoffmann este JOGGING.