EXAMEN STRUCTURI DE DATE

Subiectul 6

1 Se consideră matricea **A ∈ M(2;R)**

$$A=\begin{pmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

Se cere:

0,2p a) Ce cantitate de memorie este necesară pentru memorarea ei dacă sizeof(double)=6 ?

0,4p b) Dacă considerăm că A este rară şi o memorăm folosind clasa **SMatrix**, cum arată această memorare şi ce cantitate de memorie se foloseşte pentru stocarea ei dacă sizeof(int)=2 ?

0,4p c) Comparati cele doua rezultate de la pct a) si b) si interpretati-le.

2 Fie **B** un arbore binar cu **n** noduri în care toate nodurile sunt numai de grad 0 sau 2. Să se arate că:

1,0p a) **n** este impar;

1,0p b) Dacă **L(B)** şi **R(B)** sunt fii stâng şi drept ai lui **B** şi sunt nevizi (nenuli), atunci **L(B)** şi **R(B)** au şi ei un număr impar de noduri în componenţa fiecăruia (i.e. **|L(B)|** şi **|R(B)|** sunt impare).

3 Structura de date **Graf** memorat sub formă de liste de adiacenţă:

0,2p • exemplu

0,8p • clasa C++ (declaraţia operaţiilor),

1,0p • implementarea metodei de afişare graf.

4 Sortarea rapidă (**Quicksort**):

0,6p • exemplu, descriere

1,0p • clasă C++(folosind Templates),

0,6p • ordin de operaţii (fără calcule).

5 Ce returnează următorul cod?

1,0p

```
function mister_1(n)
r:=0;
for i:=1 to n-1 do
   for j:=i+1 to n do
      for k:=1 to j do
         r:=r+1
return r;
```

5 Pentru un arbore binar, scrieţi o metodă

1,0p

unsigned int BTree::getLegNull() **const**;

care returnează numarul total de legaturi nule din arbore.

TOTAL: 9p+1p oficiu=10p

✓

① a) 9 * sizeof(double) = 9·6 = 54.

b) matricile rare se memorează astfel

$L = \{$ (nr. linii, nr. coloane, nr. elem. nenul), (li, ci, elem. nenul) $\}$

poziția și elem. nenul.

în cazul nostru avem

$L = \{ (3,3,2); (2,3,1); (3,3,1) \}$

în acest caz avem $3 \times 3 = 9$ locații de memorie

$9 \times$ sizeof(int) = 18

②

a) $n = n_0 + n_1 + n_2 = n_0 + n_2 = n_2 + 1 + n_2 = 2n_2 + 1$ este impar

b)

④ Sortarea rapidă QuickSort.

Presupunem că avem de sortat crescător elementele unui tablou tab de tip int cu n elemente.

Pentru aceasta procedăm astfel.

Etapa 1 Alegem în mod arbitrar un element al tabloului tab și-l atribuim variabilei x

Etapa 2 Parcurgem tabloul și vom face permutări de elemente dacă este necesar ca toate elementele tabloului mai mici decât x. să fie în stânga tabloului și elementele mai mari decât x să fie în dreapta tabloului.

Etapa 3 sortăm elementele din stânga lui x și apoi elementele din dreapta lui x.

Algoritm:

1. Se alege un element arbitrar din tablou caro se atribuie lui x;
2. Se parcurge tabloul de la stânga la dreapta până se găsește un element tab[i] >= x
3. Se parcurge tabloul de la dreapta la stânga până se găsește un element tab[j] <= x.
4. Se permută între ele elementele tab[i] cu tab[j] dacă i nu l depășește pe j
5. Se continuă pașii 2, 3, 4 până când i > j: în acest moment tabloul este divizat în două
6. Pași de mai sus se continuă apoi asupra fiecăreia dintre cele două părți: până când fiecare parte conține un singur element în acest moment tabloul are elemente sortate crescător.

Exemplu:

6, 5, 4, [3] 7, 2, 9, 1, 0

I) inf = 1
sup = 8
alegem pivot = tab[4] = 3. = x
i = inf. 6 = tab[1] >= x ← i = 1
j = sup 2 = tab[6] <= x ← j = 6 => permutam pe tab[i] cu tab[j]

=> 2, 5, 4, [3] 7 6 9 1 0.

II) i++ => i = 2
j-- => j = 5 i < j => tab[2] = 5 > 3.
tab[5] = 7 > 3.

III) i++ => i = 3 => i < j => tab[3] = 4
j-- => j = 4 tab[4] = 3.) se permută.

2, 5, [3] 4 7 6 9 10.

```
void quicksort(int tab[], int prim, int ultim)
{
int min, max, separator_lista, aux;
min = prim;
max = ultim;
separator_lista = tab[(prim+ultim)/2];
do
{
while(tab[min] < separator_lista)
        min++;
while tab[max] > separator_lista)
        max--;
if (min < max) // permută
        {aux = tab[min];
        tab[min++] = tab[max];
        tab[max--] = aux;
        }
}
while(min <= max);
if (prim < max) // pentru că a ieșit din while
    quicksort(tab, prim, max);          atunci când max < min.
if (min < ultim)
    quicksort(tab, min, ultim);
}
```

Pentru această metodă de sortare timpul mediu este de ordin logaritmic

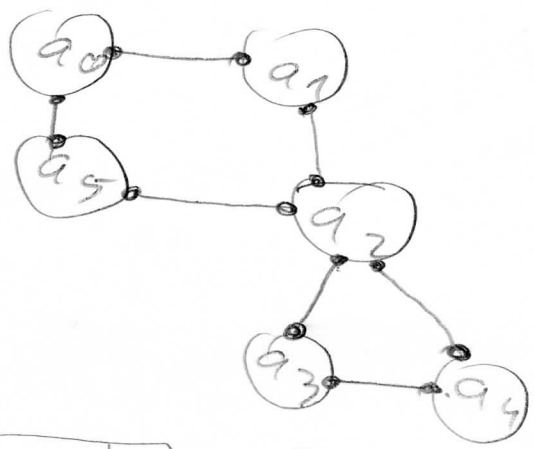
$O(n \log_2 n)$

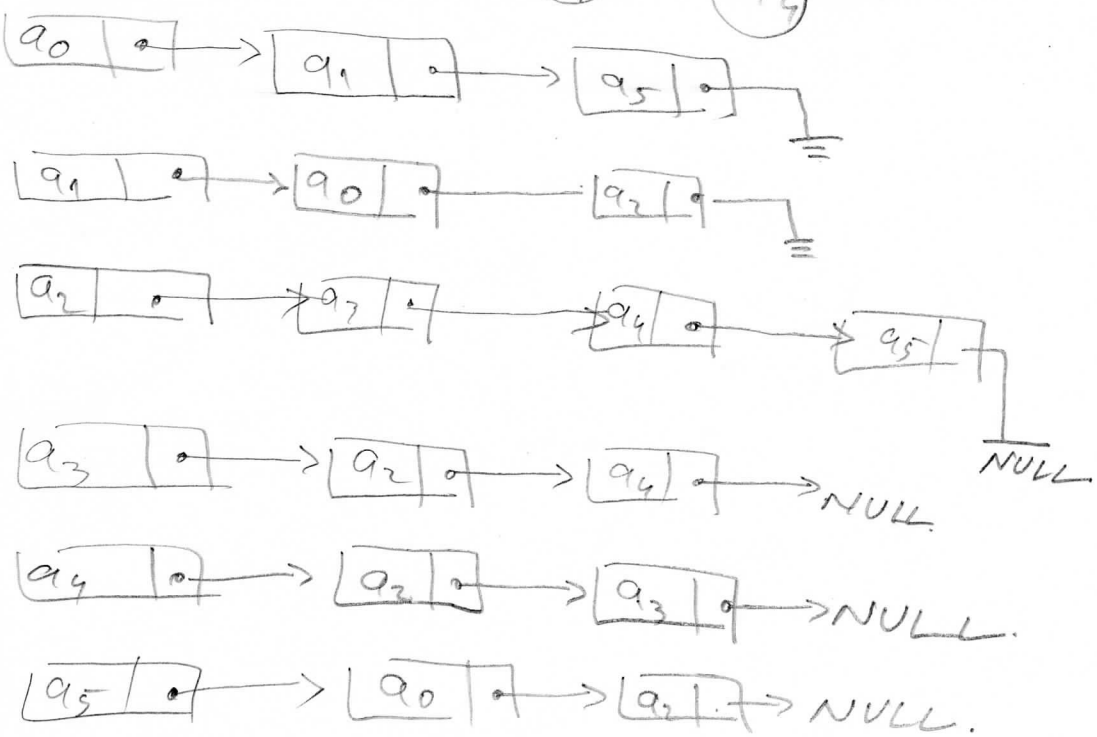
Pentru această metodă de sortare timpul mediu este logaritmic de ordinul $O(n \log_2 n)$

(3) Fie $G = (V, E)$ unde $V = \{a_0, \dots, a_{n-1}\}$.

Pentru a memora un graf se construiește un vector unidimensional de dimensiune n (nr. de vârfuri) în care elementul de pe poziția k este capul unei liste de noduri adiacente cu k.

Ex.





Structura de date in limbajul C++ derivata din clasa Graph este:

```
template <class Vertex>
class GraphAsLists : public virtual Graph
{
protected:
Array <vertex> varfuri;
Array < Lista_inlantuita <vertex>> adiac_list; // tablou de liste
public:
...
GraphAsLists (unsigned int); // constructor
void PrintConexComponent(); // printeaza comp. conexe
};
```

Pentru afișare vom supraincarca operatorul de afișare la stream-uri:

```
template <class Vertex>
ostream& GraphAsLists:: operator<< (ostream& ostr)
```

(5) Codul returnează următoarea sumă:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{j} 1 = \sum_{i=1}^{n-1} \frac{(n-i)(n+i+1)}{2} =$$

$$= \frac{(n^2+n)(n-1)}{2} + \frac{n^2}{2} - \frac{n(n+1)(2n+1)}{12} + \frac{n(n-1)}{4}$$

$= \dots$

$n=4 \Rightarrow \frac{20 \cdot 3}{2} + \frac{16}{2} - \frac{4 \cdot 5 \cdot 9}{12} - 3 = 30 + 8 - 18 = 20$ ✓

int getLengthNull( ) const

~~{ if (~~ *this

parcurgerea la dreapta