EXAMEN STRUCTURI DE DATE

Subiectul 2

1. Se consideră mulţimea **M** memorată ca secţiune a lui **N**(clasa **BitSet**) pe intervalul [0,7].

0,5p a) Dacă **M** are mulţimea suport (vectorul **data**) un singur octet de tip **unsigned char**, egal cu 00100110_2 atunci care este mulţimea M?

0,5p b) Ce octet suport corespunde multimii {1,2,3,4}?

2. Un tip particular de matrice pătratică de ordinul n este matricea bandă de lăţime 2, de forma:

2p

$$\begin{bmatrix} x & & & & \\ x & x & & O & \\ & x & x & & \\ & & \ddots & \ddots & \\ & O & & x & x \end{bmatrix}$$

Scrieţi o formulă matematica Adr:$\{1,2,...,n\}^2 \rightarrow \{1,2,...,2n-1\}$ care determină o reprezentare optimă a acestei matrici, memorând numai elementele nenule.

3. Structura de date stiva:

- 0,2p definiţie
- 0,2p exemplu
- 0,6p structura de date $\mathbf{SD_{STACK}}$ (functii + axiome)
- 0,4p declaratia **C++** pentru interfaţa (clasa virtuală) **Stack** (folosind Templates),
- 0,6p declaratia pentru implementarea sub formă de listă simplu înlănţuită (**StackAsLinkedList**).

4. Sortarea prin inserţie:

- 0,4p exemplu,
- 0,4p clasa de sortare **C++,**
- 0,2p ordin de operaţii(fara calcule)

5. Fie $\mathbf{n}=\mathbf{2}^k$, $k > 0$ şi graful **G**=(**V**, **E**), |**V**| =**n**, **V**={0, 1, ..., **n**-1} şi $(x, y) \in E \Leftrightarrow (\exists)\ 0<=i<=k-1$ a.î. $|x-y|=2^i$.

1,0p a) Să se arate că **G** este conex.

0,5p b) Câte muchii are **G** ?

0,5p c) Pentru n=3 reprezentaţi memorarea lui **G** sub formă de liste de adiacenţă.

6. Pentru un arbore binar, scrieţi o metodă:

1,0p

```
int BTree::getNumDegree0();
```

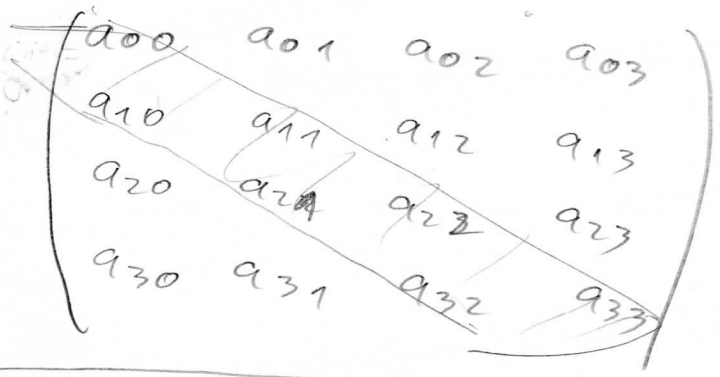
care determină numărul de noduri de grad 0 din arbore.

TOTAL: 9p+1p oficiu=10p

Subiectul 2.

(1) ✓

(2)



a_{00}	a_{10}	a_{11}	a_{21}	a_{22}	a_{32}	a_{33}

$a_{ij} = \alpha + 1 + 2(i-1)$ $\quad i \neq j \quad (j = i-1)$

$a_{ij} = \alpha + 1 + 2(i-1) + 1 = \alpha + 2i$ $\quad i = j$

Deci $a_{ij} = \alpha + i + j$.

(3) Stiva este o listă ordonată în care toate operațiile de inserare sau de ștergere se fac pe la un singur cap numit vârful stivei (TOP).

Are o structură LIFO.

Utilizări ale stivei:

- forma poloneză
- apelul subprogramelor: la fiecare apel este pusă automat pe stivă adresa de revenire (de unde trebuie să se reia execuția programului după terminarea funcției apelate.

$SD_{STACK} = \{D, d, F, A\}$.

$D = \{T, STACK, BOOLEAN\}$ (imaginile funcțiilor din F)

d = STACK (notat ca și structura de date)

FUNCȚIILE

1). creare — CREATE() ⟶ STACK

2) adăugare — PUSH(T, STACK) ⟶ STACK

3) șterge — DELETE(STACK) ⟶ STACK.

4) returnează vârful — GETTOP(STACK) ⟶ T

5) extragere un element. — POP(STACK) ⟶ T.

6) testează stiva — ISEMTS(STACK) ⟶ BOOLEAN.

AXIOME

ISEMTS(CREATE) = TRUE

ISEMTS(PUSH(i,s)) = FALSE

DELETE(CREATE) = ERROR

POP(PUSH(i, S)) = i ✓

GETTOP(CREATE) = ERROR.

GETTOP(PUSH(i,S) = i ✓.

POP(CREATE) = ERROR

PUSH(POP(S), S) = S. ✓

?? Declarația pentru interfață este:

```
#include "container.h" //
template<class T>
class Stack : public virtual container
{
public:
virtual T& getTOP() const=0. funcție virtuală pură
virtual T& pop() = 0          ——//——
virtual void push(T const&)=0;   ——//——
};
```

Stiva poate fi reprezentată ca o listă liniară înlănțuită. Pentru aceasta avem nevoie de o clasă Element cu care modelăm nodurile listei și o clasă Linklist.
(Vezi programul de la inserarea listelor)

(4) Sortarea prin inserție.

Este o sortare cu timpul de operații pătratic. Se pornește de la o listă V[0], ... V[K-1] deja sortată și caută să insereze noul element V[K] pe poziția i corespunzătoare lui deplasând elementele de pe pozițiile V[i], ... V[K-1] cu o poziție la dreapta

INSERȚIE D(n, A) (A se transmite prin referință)

```
READ A[0]
FOR j=1 TO n-1 DO
    READ A[j]
    AUX = A[j] // AUX se folosește pentru o eventuală interschimbare
    K = j-1 // K este indicele cu care se parcurge mulțimea {a1, a2, ..., aj-1}
    WHILE (AUX < aK) AND (K >= 0) DO
        aK+1 = aK // deplasează pe aK în dreapta
        K = K-1
    END
    a[K+1] = AUX // pune elementul pe poziția lui
END
```

CODUL în C (vezi calculatorul)
Pentru normală
$T(n) = 9n^2 - 176B$ unde B este numărul care indică de câte ori se efectuează cel mai interior bloc. Avem că $B \leq \frac{n(n-1)}{2}$
Deci $T(n) = O(n^2)$.

5 a)

Fie $(x,y) \in \{0,1,\dots n-1\} \times \{0,1,2,\dots,n-1\}$.

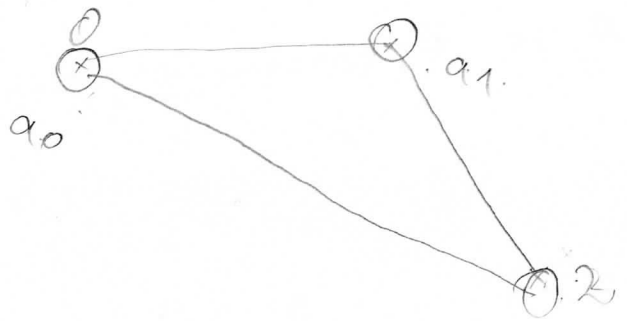
Observăm că dacă $y = x+1$ atunci $(\exists)\, i = 0 \cdot a_i$.

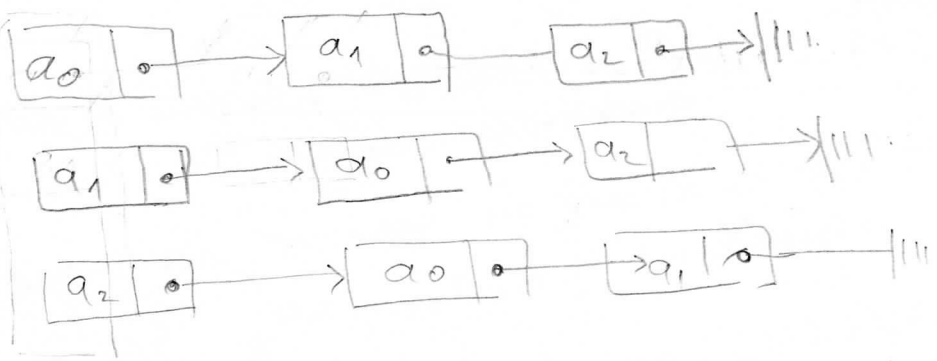
$|x-y| = 2^i$

Pe baza acestei observații rezultă că pentru orice $x, y \in \{0,1,\dots n-1\}$ există un drum (cel care unește două elemente consecutive) care unește pe x cu y.

b)

c)





⑥

```
int BTree::getNumDegree0(BTree *bt)
{
 if (bt!=NULL)
    {
     if ((bt->st==NULL)&&(bt->dr==NULL))
                    return 1;
     else
          return getNumDegree0(BTree*bt)+
                 getNumDegree0(BTree*bt);
    }
}
```

implementare pentru sortarea prin insertie.

```
template <class T>
class insertionSorter : public Sorter<T>
{
public
void Sort(Array<T>& v)
    {
    int k, j;
    T Aux;
for (j=1; j<v.getLenght(1); j++)
        {
        Aux = v.[j]; // se memorează A[j] pentru
                     cazul în care aceste se
                     schimbă
        k=j-1.
        while (AUX < V[K]) && (K>=0)
            {
            V[K+1]=V[K]; // deplasare la dreapta
            K--;
            }
        V[K+1]=AUX; // pune elementul pe poziția
        }                lui normală
    }
}; // sfârșitul clasei
```

Obs SortArray poate fi și o metodă în clasa Array<T>

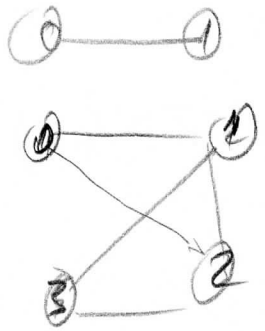
$n = 2^k$

$k=1 \Rightarrow n=2$

$k=2 \Rightarrow n=4$

- $i=0$ 2
- $i=1$ 2

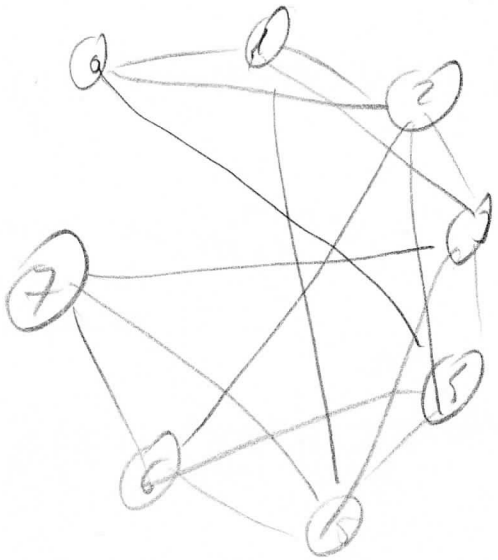
5



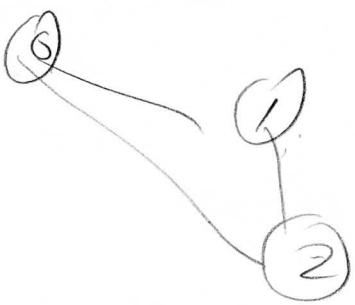
$k=3 \Rightarrow n=8$

- $i=0$ 7
- $i=1$ 6
- $i=2$ 4

17



$$\sum_{i=0}^{k-1}$$



$2^1 - 2^0 = 1$

$2^2 - 1$

$2^2 - 2^1$

$i=0 \quad n-1$

$i=1 \quad n-2^1$

$i=2 \quad n-2^2$

$$\sum_{i=0}^{k-1} 2^k - 2^i$$

$$\sum_{i=0}^{2} 2^k - 2^i =$$

$(8-1) + (8-2) + (8-4)$

$$\sum_{i=0}^{k-1} 2^k - 2^i$$

$$k \cdot 2^k - \frac{2^k - 1}{2-1}$$

$2^k(k-1)+1 = 5 \checkmark$