1. Tipuri de date

f. Cum se poate face referirea unui membru al unei variabile structură

Referirea unui membru al unei variabile structură se face:

(a) direct;

(b) indirect.

direct - folosind operatorul de selecţie

indirect - structurile sunt frecvent referite prin intermediul unei variabile pointer

h. In ce situatii este indicată transmiterea structurilor prin valoare

Transmiterea structurilor prin valoare este indicată în două situaţii:

- structura este foarte mică (aproximativ aceeaşi dimensiune ca un pointer)
- se doreşte garantarea faptului că funcţia apelată nu modifică structura transmisă.

3. Liste

a. Definitia listelor ordonate liniar

Listele ordonate liniar sunt structuri de date alcătuite dintr-o mulţime A={A_1, A_2, …, A_n }de elemente (de obicei identice), între care există o relaţie determinată de poziţia lor relativă. Astfel, fiecare element A_k are un predecesor A_{k-1} şi un succesor A_{k+1} (mai puţin elementele prim şi ultimul).

b. Cum se poate realiza reprezentarea în memorie

Reprezentarea în memorie se poate realiza:
secvenţial;
prin înlănţuirea elementelor.

c. Cum sunt dispuse elementele in reprezentarea secventiala si comparatia acesteia cu un tablou

În acerst tip de reprezentare, elementele sunt dispuse succesiv, într-o zonă contiguă de memorie. Reprezentarea este similară cu cea a unui tablou

d. Avantajele si dezavantajele reprezentarii secvenţiale

Avantaje: accesul la oricare element din listă, parcurgerea listei şi adăugarea unui element la sfârşitul listei se fac cu uşurinţă.

Dezavantaje:

- inserarea, ştergerea de elemente din mijlocul listei sunt operaţii costisitoare, deoarece presupun deplasarea unor elemente.
- dacă numărul componentelor variază în limite largi în timpul execuţiei programului, memoria nu este utilizată eficient, deoarece spaţiul alocat pentru tablou (static sau dinamic) este fix şi dimensiunea alocată trebuie să fie acoperitoare.

e. Clasificarea listelor in funcţie de locul de acces la liste

În funcţie de locul de acces la liste

- Lista FIFO (First in First Out) - inserarea se face la un capăt al listei (la sfârşit),extragerea de la celălalt capăt (din faţă).

Lista LIFO (Last in First Out) - (stivă) - operaţiile de inserare / extragere se fac de la acelaşi capăt al listei, numit vârful stivei.

f. Cum se realizează lista FIFO în reprezentare secvenţială si relatia de calcul reprezentativa

Lista FIFO se realizează în reprezentare secvenţială (folosind un tablou) ca un "tampon circular",

Folosind notaţiile:

prim = indexul primului element

ult = indexul ultimului element

ncrt = numărul de elemente din listă

n = numărul de elemente din tablou

ult = (*prim* + *ncrt*) modulo *n*

g. Cum se controlează operaţiile de inserare/extragere in cazul listelor FIFO

Operaţiile de inserare/extragere se controlează cu două variabile în două variante:

I. -poziţia (indexul) în tablou de unde se face extragerea (primul element din listă);

-poziţia unde se va face înscrierea (după poziţia ultimului element din listă).

II. -poziţia (indexul) în tablou al primului element din listă;

-numărul de elemente din listă la un moment dat.

După fiecare extragere, *prim* este incrementat modulo *n*. La orice operaţie, *ncrt* este actualizat.

h. Ce trebuie verificat înaintea unor operaţii de extragere si de adăugare

Înaintea unei operaţii de extragere trebuie verificat dacă există elemente în listă deci ncrt > 0.

Înaintea unei operaţii de adăugare se verifică dacă există spaţiu în tablou, deci dacă ncrt < N.

4. Liste înlănţuite

a. Definitia listelor inlantuite

Definiţie: O listă înlănţuită este alcătuită din noduri cu structura date şi legături în care:

- câmpul DATE reprezintă informaţia propriu zisă (un element al listei);
- câmpul LEGĂTURĂ reprezintă informaţia de secvenţă, legătura spre elementele adiacente din listă.

b. Avantajele si dezavantajele listelor inlantuite

Avantaje:

- inserarea şi ştergerea unui element se reduc la modificări ale legăturilor elementelor vecine;
- spaţiul de memorare poate fi alocat individual, pentru fiecare element al listei;
- reordonarea unei liste nu mai necesită transferuri memorie-memorie pentru interschimbarea elementelor.

Dezavantaje:

- ocuparea unui spaţiu mai mare de memorie, pentru informaţia de secvenţă;
- căutarea unui element al listei se face tot secvenţial.

c. Ce este o lista simplu inlantuita

Listă simplu înlănţuită - conţine noduri în care este specificată legătura (adresa) spre elementul următor.

d. Definitii: lant si lista circulara

O listă înlănţuită în care legătura ultimului element are valoarea NULL, care marchează sfârşitul listei, se numeşte lanţ. Dacă legătura elementului final specifică primul element se obţine o listă circulară.

e. Ce neajuns remediaza listele dublu inlantuite

Există operaţii precum înserarea înaintea elementului curent sau ştergerea elementului curent, care presupun referirea elementului dinaintea elementului curent, operaţie care se realizează cu dificultate. Pentru a remedia acest neajuns se utilizează liste dublu înlănţuite.

f. Ce este o lista dublu inlantuita

Listă dublu înlănţuită - conţine noduri în care se specifică legături către nodul precedent şi către nodul următor, oferind o mai mare flexibilitate.

5. Operatii cu liste

a. Cum se clasifica operatiile cu liste dpdv al gradului de complexitate

După gradul de complxitate, operaţiile cu liste se pot clasifica în:
Operaţii primitive:
Operaţii de caracterizare:
Operaţii complexe:

b. Enumerati operatii primitive

adăugarea unor elemente noi la sfârşitul listei;
înserarea unor elemente noi în orice loc din listă;
ştergerea unor elemente din orice poziţie a listei;
modificarea unui element dintr-o poziţie dată;
iniţializarea unui liste ca o listă vidă.

c. Enumerati operatii complexe

separarea unei liste în două sau mai multe liste;
combinarea a două sau mai multe liste în una singură;
concatenarea (alipirea);
interclasarea (rearanjarea conform cu un criteriu).
ordonarea unei liste după valorile (crescătoare sau descrescătoare) ale unei chei;
selecţia elementelor dintr-o listă care satisfac la unul sau mai multe criterii, dintr-o nouă listă.

d. Enumerati operatii de caracterizare

- determinarea lungimii listei (numărul de elemente);
- localizarea elementului din listă care îndeplineşte o anumită condiţie;

6. Arbori

a. Definitie arbori si necesitatea organizarii pe baza ierarhiei de componente

un arbore A este fie vid, fie format dintr-un nod rădăcină (R) , căruia îi este ataşat un număr finit de arbori. Aceştia sunt denumiţi subarbori ai lui A, datorită relaţiei de “subordonare” faţă de rădăcină.
Organizarea liniară de tip listă nu este întotdeauna cea mai adecvată pentru unele aplicaţii. Astfel, dacă trebuie să descriem structura unui produs, de cele mai multe ori nu prezentăm o listă a tuturor componentelor, ci utilizăm o descriere ierarhică.

b. Tipuri de noduri; inaltimea unui arbore

pentru a exprima relaţiile directe între nodurile unui arbore, se utilizează termenii tată, fiu şi frate

Înălţimea unui arbore se poate defini ca maximul dintre nivelurile nodurilor terminale, deci înălţimea unui arbore nevid este cel puţin 1 (în cazul în care arborele are un singur nod).

c. Definitii arbore binar si arbore multicai; numarul de noduri dintr-un arbore si in particular arbore binar

Dacă toate nodurile dintr-un arbore au cel mult doi descendenţi direcţi (fii), atunci arborele este denumit arbore binar

În cazul in care ordinul nodurilor nu este limitat, arborele este denumit *arbore multicăi*.

Numărul de noduri dintr-un arbore se situează între limitele determinate de ordinul şi înălţimea acestuia. Astfel, în cazul unui arbore binar de înălţime H, numărul de noduri N este cuprins între H si 2^H-1. Rezultă că înălţimea unui arbore binar cu N noduri poate varia între $\log_2 N$ şi N.

e. Care sunt modurile in care se poate parcurge un arbore

parcurgerea arborelui se efectuează în lăţime (lărgime), si in adâncime.

f. Care sunt modurile particulare de parcurgere in adancime a unui arbore; detaliati referindu-va la nodurile specifice

informaţia specifică nodului rădăcină este prelucrată (tipărită) înaintea informaţiilor din celelalte noduri ale

(sub)arborelui. Acest tip de parcurgere este denumit *parcurgere în preordine*.

. Deoarece numărul de persoane angajate într-un compartiment poate fi calculat (şi eventual tipărit) numai atunci când se cunoaşte numărul de angajaţi din toate compartimentele subordonate, rezultă că prelucrarea de la nivelul nodului rădăcina se face după prelucrarea restului arborelui respectiv, deci *parcurgerea* se realizează *în postordine*.

g. Detaliati cazul parcurgerii în lăţime a arborilor

În *cazul parcurgerii în lăţime*, se tipăreşte mai întâi informaţia din nodul rădăcină, după care sunt prelucrate, de la stânga spre dreapta, nodurile aflate pe primul nivel, apoi cele aflate pe al doilea nivel, etc. Pentru a realiza parcurgerea, se poate folosi o coadă, iniţializată cu nodul rădăcină. cât timp există noduri în coadă, se repeta următoarele operaţii:

1. este extras primul nod din coadă;
2. nodul extras este prelucrat;
3. fiii săi sunt adăugaţi în coadă, în vederea parcurgerii ulterioare.

h. Detaliati cazul parcurgerii în adancime a arborilor

În *cazul parcurgerii in adâncime*, fiii unui nod sunt vizitaţi tot de la stânga spre dreapta, dar trecerea de la fiul curent la fratele din dreapta se realizează numai după tratarea tuturor descendenţilor fiului curent (deci a întregului subarbore dominat de acesta).

Pentru a memora informaţiile relative la punctul de revenire (nodul tata şi următorul fiu neprelucrat al acestuia) se utilizează o stivă, iniţializată cu perechea:

[rădăcina, primul fiu al rădăcinii].

Cât timp stiva nu este vidă, din vârful său se extrag informaţiile despre nodul curent şi următorul fiu neprelucrat al acestuia, care devine "nod prelucrat".

Dacă nodul curent mai are şi alţi fii, atunci în stiva se introduce perechea:

[nod curent, următorul fiu neprelucrat].

De asemenea, dacă nodul prelucrat nu este nod terminal, atunci în stiva trebuie inserată şi perechea:

[nod prelucrat, primul fiu al nodului prelucrat].

7. Grafuri

a. Definitii graf, arc, cale

Printr-un *graf* se înţelege o mulţime de noduri (numite şi vârfuri) şi o aplicaţie definită pe această mulţime cu valori în aceeaşi mulţime, care face legătura între aceste noduri, legături numite *arce*, care pot fi sau nu orientate.

O *cale* este o succesiune de noduri aleasă astfel încât să existe arce care să reunească nodurile respective. O cale este simplă dacă toate nodurile, cu excepţia primului şi ultimului, sunt distincte între ele.

b. Definitii graf aciclic, digraf, graf neorientat

Un graf *aciclic* este un graf care nu conţine nici o cale de la un nod la el însuşi.

Un *graf orientat* sau *digraf* (prescurtare de la directed graf) G=(V,E) constă deci într-o mulţime V de vârfuri (sau noduri) şi o mulţime E de arce. Un arc poate fi privit ca o pereche ordonată de vârfuri (v,w) unde v este baza arcului iar w este vârful arcului. Se spune că w este adiacent lui v.

Un graf neorientat sau graf G=(N,R) este alcătuit dintr-o mulţime de noduri N şi o mulţime R de muchii. O muchie este atunci o pereche ordonată de noduri (v,w)=(w,v).

Un graf este echivalent cu un digraf în care pentru fiecare arc (v,w) există şi perechea lui (w,v).

c. Definitii graf conex, componenta conexa, graf biconex

Un graf este *conex* dacă oricare două noduri ale sale sunt conectate.

O componentă conexă a unui graf G este un subgraf conex indus maximal. Un graf conex aciclic este un arbore numit arbore liber. El poate fi convertit într-un arbore orientat alegând un nod oarecare drept rădăcină şi înlocuind fiecare muchie cu un arc orientat dinspre rădăcină.

Un graf conex G=(N,R) este *biconex* dacă prin omiterea oricărui nod v din N se obţine un graf G'=G-v conex. O componentă biconexă a unui graf este un subgraf maximal conex. Un nod v din N este punct de articulare al lui G dacă G-v nu este conex.

d. Moduri de reprezentare mai des utilizate pentru grafuri si cum trebuie sa fie făcută alegerea unuia dintre ele

Pentru grafuri există două moduri de reprezentare mai des utilizate: matricea de adiacenţe şi listele de adiacenţe.

Alegerea uneia dintre ele trebuie fi făcută în funcţie de frecvenţa operaţiilor de acces la nodurile şi muchiile grafurilor.

e. Avantajele si dezavantajele reprezentarii prin matrice de adiacenţe

Reprezentarea prin matrice de adiacenţe permite un acces rapid la arcele (muchiile) grafului fiind utilă în algoritmii în care se testează prezenţa sau absenţa unui arc oarecare. Ea este dezavantajoasă dacă numărul de arce este mai mic decât n × n, caz în care memoria necesară pentru a păstra matricea este folosită ineficient.

f. Avantajele si dezavantajele reprezentarii prin liste de adiacenţe

Reprezentarea prin liste de adiacenţe foloseşte mai bine memoria, dar determină o căutare mai anevoioasă a arcelor. În această reprezentare, pentru fiecare nod se păstrează lista arcelor către nodurile adiacente.

g. Criteriul si modalităţile de explorare a grafurilor; explicati pe scurt fiecare dintre aceste modalitati

trecerea de la un nod la altul nu se poate face oricum, ci doar de-a lungul unui arc direct (muchii directe) dintre ele.

După ordinea de explorare a arcelor se cunosc două modalităţi de explorare: în lărgime şi în adâncime

a. Explorarea în lărgime

La explorarea în lărgime, după vizitarea nodului iniţial, se explorează toate nodurile adiacente lui, se trece apoi la primul nod adiacent şi se explorează toate nodurile adiacente acestuia.

b Explorarea în adâncime

La explorarea în adâncime se marchează vizitarea nodului iniţial după care se parcurge în adâncime, recursiv, fiecare nod adiacent cu el. După vizitarea tuturor vârfurilor ce pot fi atinse din nodul de start parcurgerea se consideră încheiată. Dacă rămân noduri nevizitate se alege un nou nod de start şi se repetă procedeul.

8. Backtracking

a. Cum se mai numeste metoda Backtracking; justificati utilizarea acestei metode - in general ineficiente - in contextul modularizarii

Metoda backtracking se mai numeste modularizare. Metoda Backtracking (a căutării cu revenire), în general ineficientă, având complexitate exponenţială, poate fi utilizată la optimizarea procesului de căutare, evitând căile care nu duc la o soluţie.
În multe aplicaţii, găsirea soluţiilor este rezultatului unui proces de căutare sistematică, cu încercari repetate şi reveniri în caz de nereuşită. Dintre problemele ale caror soluţii se găsesc prin căutare menţionăm câteve exemple:

- găsirea unei căi de ieşire dintr-un labirint;
- plasarea pe o tablă de şah a opt dame care nu se atacă între ele. găsirea unui traseu care acoperă tabla de şah, generat adoptând săritura calului fără a trece de două ori prin aceeaşi poziţie.

9. Algoritmi de cautare

a. Ce este o regula euristica; justificati utilizarea acestei metode in cazul algoritmilor de cautare

O regulă euristică oferă o metodă de rezolvare a problemelor, sau o metodă de căutare. Ea nu dă rezultate corecte la orice moment de timp şi nu este garantată că găseşte cea mai bună soluţie, dar în acelaşi timp poate reduce timpul de căutare. Metodele de acest tip pot fi folosite pentru reducerea mărimii arborilor de căutare în cazurile arborilor foarte mari.

b. Câteva dintre avantajele strategiei de căutare înapoi

Prin aplicarea regulilor de descompunere problema se transformă în subprobleme de complexitate mai mică.

Aceasta poartă şi denumirea de control dirijat prin obiectivul problemei sau control de sus în jos. Datorită caracteristicilor sale, această metodă se mai numeşte şi *reductivă*.

Avantaje:

- atunci când sunt necesare sau când toate posibilităţile au fost explorate, sistemul recurge la întrebări adresate utilizatorului;
-

- arborele de căutare este adesea mai puţin adânc decât cel aferent strategiei de căutare înainte;
- procesul de raţionare este interactiv.

10. Algoritmi pentru minimizare cai

a. Rezolvarea problemei căii de cost minim de la sursă la destinatar prin algoritmul Dijkstra; tehnica folosita, in ce consta si ce structuri de date utilizeaza

Rezolvarea acestei probleme se bazează pe o tehnică “greedy” datorată lui E.W. Dijkstra. Ea constă în păstrarea unei mulţimi ***Selectate*** de vârfuri ale căror distanţe minime faţă de ***sursă*** sunt cunoscute. Iniţial, ***Selectate*** conţine doar vârful sursă; la fiecare pas, se adaugă la ***Selectate*** un vârf a cărui distanţă faţă de un vârf din ***Selectate*** este minimă.

În rezolvare se utilizează:

- un tablou ***Distanţă*** al distanţelor minime de la sursă la fiecare vârf.
- o matrice ***Cost*** de costuri, în care Cost [i, j] este costul asociat arcului (i,j); dacă nu există un arc (i,j), atunci se consideră pentru Cost [i, j] o valoare *infinit* (practic, foarte mare).

b. Cum se mai numeste metoda Kruskal si pe ce se bazeaza algoritmul Kruskal

Metoda arborelui minim de acoperire. Algoritmul construieşte treptat mulţimea T a muchilor arborelui minimal adăugând la fiecare pas muchia care nu formează cicluri cu muchiile aflate deja în T.

c. Asemanarea esentiala si deosebirea esentiala intre algoritmii Prim si Kruskal

Un alt algoritm greedy pentru determinarea arborelui parţial de cost minim ale unui graf se datorează lui **Prim** (1957).

Deosebirea constă în faptul că, la fiecare pas, mulţimea A de muchii alese împreună cu mulţimea U a vârfurilor pe care le conectează formează un subarbore de cost minim pentru subgraful (U,A) al lui G (şi nu o pădure ca în algoritmul lui Kruskal).

d. Graf si componente biconexe – ce sunt; scurta explicatie

Dacă (v,w) aparţine lui R, atunci subgraful având nodurile v şi w şi muchia (v,w) este biconex. Fiecare muchie a grafului aparţine unei singure componente biconexe. Altfel spus, componentele biconexe ale unui graf induc o partiţie pe muchiile sale. Totodată, dacă există un ciclu simplu prin nodurile u şi v atunci u şi v aparţin aceleiaşi componente biconexe.

Urmărind parcurgerea unei componente biconexe, observăm că primul arc traversat este de arbore. Vârful acestuia indică un nod numit centrul componentei, deoarece joacă un rol special în găsirea componentei biconexe.

Determinarea componentei biconexe se poate termina în momentul încheierii explorării centrului său, folosind proprietăţile parcurgerii în adâncime a grafului. O componentă biconexă include centrul s, tatăl acestuia, precum şi descendenţii lui s accesibili prin căi ale arborelui de acoperire în adâncime care nu trec prin alte centre.

11. Algoritmi pentru simulare 3D

a. Modalitatile de tratare a problemei reprezentarii corpurilor 3D; detaliati grupele mari de algoritmi; care dintre grupe este mai eficient dpdv al timpului de executie?

Există două modalităţi de abordare a problemei:

- Algoritmi "spaţiu-obiect"
- Algoritmi "spaţiu-imagine"
- Primii iau în considerare faptul că la receptorul vizual ajung numai razele de lumină reflectate de porţiunile de faţetă nemascate (între acestea şi observator nu se interpune nici o altă faţetă). Deci se compară fiecare faţetă dintre cele NF existente cu toate celelalte faţete ale corpului, în scopul de a elimina acele porţiuni care nu sunt vizibile (sunt mascate). Subrutinele de comparare sunt parcurse de NF × NF ori.
- Algoritmii "spaţiu-imagine" presupun analiza pixel cu pixel a zonei de lucru din ecran pentru a determina care anume dintre faţete este vizibilă într-un anumit punct. Dacă ecranul are NP pixeli, atunci subrutinel de comparare vor fi parcurse de NF × NP ori.
- În realitate, algoritmii "spaţiuimagine" sunt mai rapizi, deoarece subrutinele de comparare individuală sunt mult mai simple şi deci consumă mai puţin timp.

b. Modelul matematic al tratarii imaginilor prin algoritmii "spaţiu-obiect"

Transpunerea matematică se bazează pe propoziţia: "Un punct este interior unui triunghi atunci când, faţă fiecare latură, el se află de aceeaşi parte a acesteia caşi vârful opus laturii respective.

Dată fiind ecuaţia dreptei prin două puncte

$(x - x_1)(y - y_1) - (x_2 - x_1)(y_2 - y_1) = 0$

Ax ——————————— xB

Mx

$E(M,A,B) = E(x_M, y_M, x_1, y_1, x_2, y_2) = (x - x_1)(y - y_1) - (x_2 - x_1)(y_2 - y_1)$

Prin SGN (E) se determină punctele obiectiv (M).

Urmează determinarea punctului mască potenţial pentru perechea (faţetă curentă, punct obiectiv). Pentru aceasta se construieşte sistemul:

$$\begin{vmatrix} x_p - x_1 & y_p - y_1 & z_p - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0$$ Ecuaţia planului care trece prin 3 puncte date

$$\frac{x_p - x_o}{x_M - x_0} = \frac{y_p - x_o}{y_M - y_o} = \frac{x_p - x_1}{z_M - z_o}$$ Ecuaţia în 3D a dreptei OM

După dezvoltare, sistemul va avea forma:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

După calcularea lui P ($x_p y_p z_p$) se vor afla distanţele OM şi OP ţinând cont de faptul că

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

urmând algoritmul precizat anterior.

c. Avantaje si dezavantaje ale *Buffer-ului de adâncime*

Sunt uşor de implementat, dar prezintă dezavantajul de a necesita memorie suplimentară pentru păstrarea valorilor cotei Z ale unor mulţimi de puncte bine detreminate.

d. Principiul algoritmilor *Linie de baleiaj (scan-line),* cand se recomanda, dezavantaje

Algoritmii de acest tip reconstituie imaginea linie cu linie, tratând ansamblul faţetelor poligonale care descriu corpul. Sunt recomandabile atunci când nu se dispune de memorie foarte mare, pentru număr mare de faţete, fiind însă mai lenţi şi performanţele lor fiind dependente de complexitatea corpului de reprezentat.

e. Metoda de reprezentare a suprafetelor in spatiu; scurta descriere a formei de reprezentare

Pentru reprezentarea în 3D se poate utiliza o metodă de reprezentare a suprafeţelor în spaţiu folosind două familii de curbe cubice corespunzătoare celor două direcţii într-un plan xOy. Prin urmare, ecuaţia suprafeţei trebuie să se exprime în funcţie de doi parametri s şi t, adică

$x = x(t,s)$, $y = y(t,s)$, $z = z(t,s)$, $s, t \in [0,1]$

Aceste relaţii sunt ecuaţiile unei suprafeţe *bicubice.*

Pentru reprezentarea imaginii suprafeţei pe ecran s-a utilizat forma de reprezentare Hermite: suprafaţa să treacă prin patru puncte din spaţiu, corespunzătoare valorilor extreme 0 şi 1 pentru parametrii s şi t (notate cu P_{00}, P_{01}, P_{10}, P_{11},) şi să aibă trei tangente la suprafaţă, date în fiecare dintre aceste puncte.