**Fişă de laborator**

**Algoritmi şi structuri de date**

|  |  |  |
| --- | --- | --- |
| **Partea I Algortimi fundamentali numerici şi seminumerici.**  **Problema sortării** |  |  |
| **Laboratorul 1 Generalităţi** |  |  |
| 1. Rezolvarea unor probleme simple ce să exemplifice erorile ce pot surveni datorită faptului ca se aplică operatori ce nu sunt definiţi corespunzător tipului de date folosit şi respectiv dacă nu se ţine cont de precedenţa operatorilor 2. Implementarea problemelor ce ţin de numărul de elemente identice într-o mulţime (1,3,4,1,1) = 3 bucăţi   [Se dau 5 valori întregi a,b,c,d şi e. Construiţi un algoritm care identifică următoarele cazuri: 1.) Există 2 valori identice, 2.) Există 2 cȃte 2 valori identice, 3.) Există 3 valori identice, 4.) Există 3valori identice şi celelalte 2 sunt de asemena identice, 5.) Există 4 valori identice şi 6.) Toate valorile sunt identice] [Să se verifice dacă valorile pot fi elementele consecutive ale unei progresii aritmetice cu raţia 1.]   1. Implementarea algoritmului de determinare a numerelor prime şi optimizările acestuia. 2. Algoritmul de determinare a cifrelor unui numǎr, 3. Algoritmul de determinare a minimului dintr-o mulţime introdusă secvenţial   [Se citesc de la tastaură o serie de numere pȃnă cȃnd se introduce o valoare dată (-1). Să se afişeze minimul valorilor citite, şi eventual numărul de apariţie a acestei valori]   1. Algoritmul de determinarea numerelor palindrom   [Un număr este palindrom dacă în reprezentarea pe cifre acesta are aceeaşi valore citit de la dreapta la stȃnga cȃt şi de la stȃnga la dreapta (11211)]   1. Algoritmul lui Euclid |  |  |
| **Laboratorul 2 Vectori statici** |  |  |
| 1. Algoritmul de calcul a sumei/produsului/normei elementelor unui vector 2. Algoritmul de calcul a minimului/maximului elementelor unui vector 3. Algoritmul de sortare pe baza funcţiei sign (--,000,+++)   [Se dă un vector de valori întregi (-,0,+). Construiţi un algoritm de sortare prin care valorile 0 ajung la mijloc între elementele negative în stȃnga şi cele pozitive în dreapta. Algoritmul se va încadra în timpul de execuţie O(n) (adică o singură parcurgere) Ex : (-3,0,2,10,0,-4,-1) > (-3,-4,-1,0,0,10,2)]   1. Algortimul de sortare bublleSort şi optimizǎri 2. Algoritmul de sortare Selection Sort 3. Algoritmul de sortare prin selecţie directǎ 4. Algoritmul de sortare prin inserţie |  |  |
| **Laboratorul 3 Vectori dinamici** |  |  |
| 1. Concatenarea a doi vectori 2. Simularea reuniunii a 2 vectori 3. Simularea intersecţiei a 2 vectori 4. Simularea diferenţei pe mulţimi a elementelor a 2 vectori 5. Algoritmul de determinare dacă un vector de dimensiune mai mică se regăseşte într-un vector de dimensiune mai mare   [Se dau 2 vectori de dimensiune n şi m, cu n<m. Să se afişeze dacă vectorul de dimensiune n se regăseşte în vectorul de dimensine m şi eventual numărul de apariţii]   1. Adunarea a două numere reprezentate ca şi vectori de cifre   [Se dau doi vectori a căror valori sunt cifrele în baza 10. Construiţi un algoritm în urma căruia să rezulte tot un vector cu valori cifre dar care reprezintă suma vectoruilor iniţiali. Ex. (9,6,1,0)+(8,4,4) = (1,0,4,5,4)]   1. Produsul a două numere reprezentate ca vectori de cifre   [Analog cu datele de intrare a problemei anterioare doar că se cere produsul a cei doi vectori. Ex. (9,6,1,0)x(8,4,4) =(8,1,1,0,8,4,0)] |  |  |
| **Laboratorul 4 Matrici** |  |  |
| 1. Construirea matricii de dimensiune n\*m a primelor n\*m numere prime   [Ex (n = 3, m = 3) ]   1. Identificarea zonelor din matrice (de o parte şi de alta a diagonalei principale şi secundare)   [Constriţi următoarele matrici pătratice de dimensiune n şi ]   1. Parcurgerea în spiralǎ a unei matrici   [Constriţi un algoritm de afişare a elementelor unei matrici astfel: se afişează în ordine elementele de pe margine după care se face abstracţie de acestea şi se repetă algoritmul.  Ex. = (11, 12, 13, 14, 15, 20, 25, 30, 35, 34, 33, 32, 31, 26, 21, 16, 17, 18, 19, 24, 29, 28, 27, 22, 23)]   1. Suma a două matrici 2. Produsul a două matrici 3. Incluziunea a două matrici |  |  |
| **Partea a II-a Tehnici de programare** |  |  |
| **Laboratorul 5 Recursivitate** |  |  |
| 1. Implementarea funcţiei factorial recursiv 2. Generarea şirului lui Fibonacci recursiv 3. Implementarea noţiunilor elementare de grafică şi construirea fractalului lui Cantor 4. Construirea fractalului lui Koch (fulg de nea) 5. Construirea fractalului lui Sierpinski 6. Implementarea algoritmului pentru rezolvarea problemei Turnurile din Hanoi cu 3 tije   [Această problemă presupune existenţa unui set de n discuri de diferite mărimi, aşezate în ordine descrescătoare pe o tijă numită sursă (discul cu circumferinţa cea mai mare se găseşte cel mai jos). Există de asemenea încă două tije numite intermediar şi destinaţie. Obiectivul problemei constă în mutarea celor n discuri de pe tija sursă pe tija destinaţie folosind tija intermediar cu următoarele trei restricţii: 1.) Se poate muta o dată o singură piesă de pe o anumită tijă (cea mai de sus). 2.) Nu se poate pune un disc de dimensiune mai mare peste un disc de dimensiune mai mică. 3.) Numărul de mutări trebuie să fie minim]   1. Implementarea algoritmului pentru rezolvarea problemei Turnurile din Hanoi cu 4 tije   [Analog cu enunţul de la problema 33 cu specificaţia că există două tije intermediare]   1. Algortimul de sortare Merge Sort |  |  |
| **Laboratorul 6 Recursivitate în plan** |  |  |
| 1. Problema platourilor de altitudine maximală într-o matrice de valori întregi   [Se dă o matrice de valori întregi. Se defineşte un platou ca şi o secvenţă de elemente din matrice, adiacente (N, V, S, E), de aceeaşi valoare.  Pentru o matrice dată să se determine prin afişarea elementelor constitutive, cel mai larg platou.  > {(1,3),(2,3),(3,3),(4,3),(4,4)}]   1. Problema teritoriilor (valori 0 adiacente înconjurate de aceeaşi valoare) (jocul de Go)   [Se dă o matrice de valori 0,1,2. Se defineşte teritoriul lui 1, un platou (definit ca în problema 36) de valoare 0, pentru care toţi vecinii sunt 1, analog, teritoriul lui 2 este un platou de valoare 0 pentru care toţi vecinii au valoarea 2. Pentru o matrice dată contorizaţi elementele constitutive ale teritoriilor lui 1 şi 2.  > (4,2)]   1. Algoritmul jocului minesweeper. |  |  |
| **Laboratorul 7 Metode aleatoare** |  |  |
| 1. Metode aleatoare de căutare în spaţiul soluţiilor, pătratul magic 2. Metode aleatoare de selecţie (Monte Carlo)   [Se consideră ca date de intrare un vector de simboluri şi ponderile de apariţie a acestora. Construiţi un algoritm care extrage un simbol din mulţimea simbolurilor astfel încȃt să ţină cont de ponderea de apariţie a acestuia]   1. Construcţia unui jucător virtual în baza selecţiei aleatoare pentru jocul mastermind 2. Algoritmul de sortare Quiq Sort |  |  |
| **Laboratorul 8 tehnica de programare „Divide et Imperra” (Divide and Conquer (DC))** |  |  |
| 1. Determinarea minimului prin DC 2. Căutarea binară 3. Ridicarea la putere a unui număr prin DC |  |  |
| **Laboratorul 9 tehnica de programare „Backtracking”** |  |  |
| 1. Generarea permutărilor unei mulţimi 2. Generarea aranjamentelor unei mulţimi 3. Generarea combinărilor unei mulţimi 4. Generarea submulţimilor unei mulţimi 5. Generarea partiţiilor unei mulţimi 6. Implementarea metodei backtracking generalizat 7. Problema damelor.   [Considerăm o tablă de şah de dimensiune N. Ne interesează toate posibilităţile de a plasa N regine pe tablă astfel încât oricum am alege două regine, acestea să nu se atace reciproc.] |  |  |
| **Laboratorul 10 tehnica de programare „Greedy” şi intropducere în algortimi genetici** |  |  |
| 1. Problema construirii unei sume   [Presupunem că trebuie să plătim o sumă de bani S şi că avem la dispoziţie un număr infinit de monede de valoare 25, 10, 5 şi 1. Se cere să se determine numărul minim de monede necesare pentru a plăti suma S.]   1. Problema spectacolelor   [Se doreşte organizarea unui număr cȃt mai mare de spectacole, într-un interval de timp oarecare Ştiind că există N artişti interesaţi să susţină un spectacol şi că fiecare artist poate să suţină spectacolul doar în intervalul de timp [st,end], determinaţi numărul maxim de spectacole care pot fi organizate astfel încât intervalele de desfăşurare a oricăror două spectacole să nu se suprapună.]   1. Implementarea operatorului genetic de mutaţie   [Fără contextul algoritmilor genetici pentru această problemă se cere construirea unui operator unar ce acţionează pe vectori, astfel încȃt prin acţiunea acestuia se iau la întȃmplare cȃteva elemenet din vectorul iniţial şi acestea se schimbă]   1. Implementarea operatorului genetic de încrucişare   [Fără contextul algoritmilor genetici pentru această problemă se cere amestecarea a doi vectori după următorul mecanism: Se aleg cȃteva puncte de tăietură iar vetorul rezultat se obţine prin combinarea celor doi vectori iniţiali copiind alternativ elementele dintre două pucnte de tăietură adiacente  Ex: (1,2,3,4,5,6,7) + (a,b,c,d,e,f,g) şi punctele de tăietură (2,5) se obţine (1,2,c,d,e,6,7)]   1. Implementarea metodei de selecţie bazată pe ordonare 2. Implementarea unui algoritm genetic pentru rezolvarea unui sistem de ecuaţii cu ajutorul algoritmilor dezvoltaţi la 56,57 şi 58 |  |  |
| **Partea a III-a Domenii conexe în algoritmică** |  |  |
| **Laboratorul 11 Introducere în geometria computaţională** |  |  |
| 1. Reprezentarea grafică a intersecţiei a două drepte 2. Construirea recursivă a poligonului dat de mijloacele laturilor poligonului anterior 3. Algortimi lui Jarvis de determinare a înfăşrării convexe 4. Algortimi lui Graham de determinare a înfăşrării convexe |  |  |
| **Laboratorul 12 Introducere în teoria grafurilor** |  |  |
| 1. Parcurgerea unui graf 2. Algoritmul lui Dijkstra 3. Algoritmul lui Bellman Ford |  |  |
| **Laboratorul 13 Introducere în tehnicile avansate de programare. Structuri de date înlănţuite definite de utilizator.** |  |  |
| 1. Implementarea tipului abstract de date listă simplu înlănţuită 2. Implementarea tipului abstract de date lista circulară 3. Problema cavalerilor mesei rotunde   [Se citeşte un număr natural N reprezentând numărul de cavaleri aşezaţi la masa rotundă. Se va începe o numărătoare începând de la cavalerul cu numărul de ordine 1, oprindu-se la cavalerul imediat următor, adică 2, care este eliminat, după aceea, se numără până la 2, oprindu-se pe al doilea cavaler după cel eliminat anterior, acesta fiind cel cu numărul de ordine 4, care este şi el eliminat. După aceea se numără până la 3, oprindu-se pe al treilea cavaler după ultimul eliminat. numărătoarea se opreşte atunci când mai rămâne un singur cavaler neeliminat.]   1. Implementarea tipului abstract de date listă dublu înlănţuită |  |  |
| **Laboratorul 14 Introducere în programarea dinamică** |  |  |
| 1. Problema labirintului. (Algoritmul lui Lee)   [Se dă o matrice pătratică de dimensiune N, cu valori de 0 sau de 1, codificând un labirint. Valoarea 0 reprezintă o cameră deschisă, iar valoarea zero o cameră închisă. Se cere cel mai scurt drum de la poziţia (1, 1) la poziţia (N, N), mergând doar prin camere deschise şi doar la stânga, dreapta, în jos sau în sus. Nu se poate trece de două ori prin acelaşi loc]   1. Problema subsecvenţei de sumă maximă   [Considerăm un număr natural N şi un vector A cu N elemente numere întregi. O subsecvenţă [st, dr] a vectorului A reprezintă secvenţa de elemente A[st], A[st + 1], ..., A[dr]. Suma unei subsecvenţe reprezintă suma tuturor elementelor acelei subsecvenţe. Se cere determinarea unei subsecvenţe de sumă maximă. Ex (-6 1 -3 4 5 -1 3 -8 -9 1) > (11)]   1. Problema rucsacului   [Se consideră N obiecte caracterizate de mărimile: greutate şi valoare. Se considerăm un rucscac de capacitate C. Ne interesează alegerea unei submulţimi de obiecte a căror greutate totală să fie cel mult C şi a căror valoare să fie maximă.] |  |  |

Algoritmii dezvoltaţi pentru rezolvarea problemelor de la 1 pȃnă la 72 inclusiv, optimizările, adnotările şi rezultatele experimentale obţinute pe baza acestora vor constitui portofoliul fiecărui student.