Structuri de Date

Examen – partea 2

1. Se da o matrice n\*n si m opratii de 2 tipuri

* Adauga x elementului de pe pozitia i j
* Care este a k-a cea mai mare linie din punct de vedere al sumei.

Rezolvare:

O rezolvare posibilă constă în determinarea sumei pentru fiecare linie (complexitate O(n2)), indicii liniilor fiind memorate ulterior într-un arbore binar de căutare. Crearea unui astfel de arbore are complexitatea de O(h), unde h este înălțimea arborelui. Pentru prima operație, adăugăm elementului de pe poziția i j x (complexitate O(1)) și actualizăm arborele de căutare în funcție de noua sumă specifică liniei i. Actualizarea are complexitate O(h). Căutarea al k-ului maxim are complexitatea O(h). Putem de asemenea să modificăm puțin arborele în așa fel încât în stânga unui nod să se memoreze valoarea mai mare decât el, iar în dreapta valoarea mai mică.

Complexitatea algoritmului descris este O(.

1. Se dau n valori. Calculati un vector cu alte n valori care sunt obtinute din scaderea fiecarui element a primului element din dreapta mai mic ca el, daca exista.

Exemplu:

[9,4,3,6,1] -> [5,1,2,5,1]

5 = 9 – 4

1 = 4 – 3

2 = 3 – 1

5 = 6 – 1

Rezolvare:

Fie v = [9, 4, 3, 6, 1].

Parcurgem vectorul cu numere și adăugăm pe stivă indicii acestora. Atunci când avem cel puțin 2 elemente pe stivă, scoatem 2 indici din vârful stivei și verificăm dacă numerele din vector aflate pe pozițiile respective respectă condiția enunțată în cerință (elementul cu indice mai mare este mai mic decât elementul cu indice mai mic). Dacă este adevărat, în vector, pe poziția corespunzătoare numărului mai mare, punem diferența dintre cele 2 numere, iar în stivă adăugăm celălalt indice. Repetăm această operație până când ajungem la o contradicție (când elementul cu indice mai mare este mai mare decât elementul cu indice mai mic). În caz contrar, le adăugăm din nou pe cele 2 în stivă și trecem la numărul următor din vector. Când mai avem un singur element în vector ne oprim. Determinând minimul pe măsură ce se parcurge vectorul, complexitatea algoritmului este O(n).

Explicații:

Adăugăm v[0] = 9

|  |
| --- |
| 0 (9) |

Adăgăm v[1] = 4

|  |
| --- |
| 1 (4) |
| 0 (9) |

Avem 2 elemente în stivă. Cum v[1] < v[0], atunci v[0] = v[0] – v[1], iar vectorul se transformă în

v = [5, 4, 3, 6, 1], iar stiva va arăta astfel:

|  |
| --- |
| 1 (4) |

Adăgăm v[2] = 3

|  |
| --- |
| 2 (3) |
| 1 (4) |

Avem 2 elemente în stivă. Cum v[2] < v[1], atunci v[1] = v[1] – v[2], iar vectorul se transformă în

v = [5, 1, 3, 6, 1], iar stiva va arăta astfel:

|  |
| --- |
| 2 (3) |

Adăgăm v[3] = 6

|  |
| --- |
| 3 (6) |
| 2 (3) |

Avem 2 elemente în stivă. Cum v[3] > v[2], trecem la următorul element din vector.

Adăgăm v[4] = 1

|  |
| --- |
| 4 (1) |
| 3 (6) |
| 2 (3) |

Avem mai mult de 2 elemente în stivă. Cum v[4] < v[3], atunci v[3] = v[3] – v[4], iar vectorul se transformă în v = [5, 1, 3, 5, 1], iar stiva va arăta astfel:

|  |
| --- |
| 4 (1) |
| 2 (3) |

Avem 2 elemente în stivă. Cum v[4] < v[2], atunci v[2] = v[2] – v[4], iar vectorul se transformă în

v = [5, 1, 2, 5, 1], iar stiva va arăta astfel:

|  |
| --- |
| 4 (1) |

Cum mai avem un singur element în vector ne oprim, rezultatul fiind:

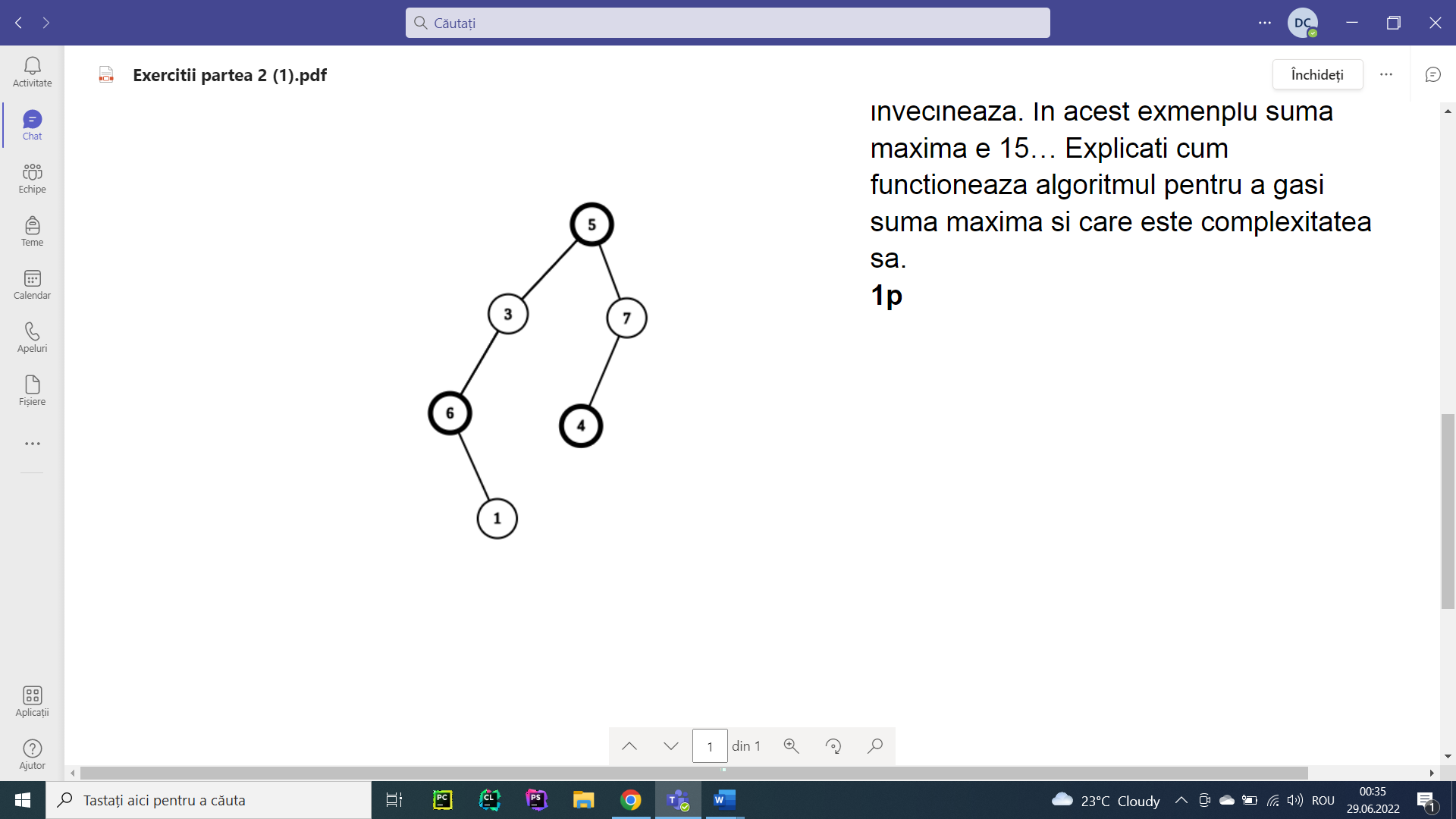
v = [5, 1, 2, 5, 1]

1. Se dau 2 vectori de n elemente. Amandoi sortati. Gasiti al k-lea cel mai mic element din reuniunea lor. Explicati algoritmul + complexitatea.

Rezolvare:

O metodă de rezolvare implică folosirea algoritmului de căutare binară care are complexitatea O(logn). Dacă k-1 este mai mare decât suma primelor jumătăți ale celor 2 vectori (k-1 > n/2 + n/2, adică k-1> n), înseamnă că nu putem găsi al k-lea minim în intervalul [0, n/2] al unuia dintre cei 2 vectori. Soluția înjumătățește vectorii până când unul dintre vectorii rămâne fără elemente. La fiecare înjumătățire scade și valoarea k-ului. Când un vector nu mai are elemente, al k-lea minim se va găsi în celălalt vector pe poziția k (un k diferit decât cel inițial).

1. Se da un arbore binar, gasiti suma maxima a unor elemente care nu se invecineaza. In acest exemplu suma maxima e 15... Explicati cum functioneaza algoritmul pentru a gasi suma maxima si care este complexitatea.



Rezolvare:

Problema poate fi abordată prin intermediul programării dinamice. Stocăm suma maximă incluzând un nod sau excluzându-l într-o matrice. Apelăm recursiv pentru nepoții nodurilor dacă nodul este inclus sau pentru vecini dacă nodul este exclus. Complexitatea algoritmului este O(n).