Universitatea din Craiova
Facultatea de Automatică, Calculatoare și Electronică

Proiectarea Algoritmilor

Raport tehnic-Temă de casă

Studentă: Ristea Gabriela-Valentina

Specializarea: Calculatoare Română
Grupa: CR 1.3A
Anul: I

Iunie 2023

Cuprins

1 Enuntul problemei

Se dă o tijă de lungime n și un tabel de prețuri pentru tijele de lungime de la 1 la n, determinați venitul maxim care poate fi obținut prin tăierea tijei și vânzarea pieselor. Rețineți că pentru o soluție optimală, există posibilitatea ca tija să nu fie taiată deloc. Implementați doi algoritmi diferiți care rezolvă această problemă. De exemplu, dacă se dă o tijă cu o lungime n = 4 și un tabel de prețuri pentru lungimi cuprinse între 1 și 4, trebuie să găsim venitul maxim care poate fi obținut din tăierea tijei.

Lungime Tijă	Preț
1	2
2	5
3	7
4	8

În acest scenariu, există două opțiuni pentru tăierea tijei: o puteți tăia în bucăți de 2 × item1 și item2 pentru a genera $2 \times 2 + 5 = 9$ un venit total, sau o puteți tăia în bucăți de 2 × item2 pentru un venit total de $2 \times 5 = 10$.

2 Abordarea problemei

Pentru rezolvarea acestei probleme am implmentat doi algoritmi care abordează două metode diferite.

Prima variantă este iterativă și utilizează programarea dinamică, scopul fiind să determine cel mai mare venit obținut prin tăierea unei tije de lungime n și vânzarea bucăților rezultate. Fiecare porțiune din tijă are un anumit preț care este salvat într-un vector.

A doua variantă este recursivă și folosește metoda problemei rucsacului, având același rezultat final ca și prima variantă iterativă.

2.1 Algoritmi utilizați

Pentru crearea seturilor de date asupra cărora se vor rula algoritmii am realizat o funție **"create-test"**. Vectorul de prețuri este creat aleatoriu, în ordine crescătoare, folosind funcția **"rand()"**.

Pseudocodul funcției

*func*tion **create-test()**
1. deschide fișierul **"nume-fisier.txt"** pentru scriere
2. **dacă** fisierul nu poate fi deschis
3. **afișează** mesajul **"Fisierul nu poate fi deschis"**
4. se iese cu codul de eroare **"1"**
5. scrie in fisierul deschis mesajul **"Dimensiunea tijei: "**
6. se initializeaza variabila **"dim_tija"** pentru dimensiunea tijei
7. se scrie in fisierul deschis aceasta dimensiune
8. **pentru** i = 0, dim_tija - 1 executa
9. **scrie** in fisier valoarea lui i + 1
10. se aloca dinamic dimensiunea unui vector de intregi pentru preturile portiunilor din tija
11. se genereaza aleatoriu o vlaoare intre 1 si 10 pentru pozitia 0 din vector
12. se scrie in fisier valoarea generata
13. **pentru** i = 1, dim_tija - 1 executa
14. genereaza o valoare pentru fiecare element din vectorul de preturi
15. scrie in fisier aceste valori

16. se elibereaza memoria alocata pentru vectorul de preturi
end *function*

Pentru a încarca datele din fișierele create în vederea prelucrării acestora, am realizat o funcție **"load-data"**.

Pseudocodul funcției

*func*tion **load_data(*l*ength[], price[], dim_tija)**
1. deschide fisierul **"nume-fisier.txt"** pentru citire
2. **dacă** fisierul nu poate fi deschis
3. **afișează** mesajul **"Fisierul nu poate fi deschis"**
4. se iese cu codul de eroare **"-1"**
5. citeste din fisierul deschis mesajul de pe prima linie si stocheaza valoarea in variabila **"dim_tija "**
6. **pentru** i = 0, dim_tija - 1 executa
7. **citeste** din fisier o valoare si o stocheaza in length[i]
8. **pentru** i = 0, dim_tija - 1 executa
9. **citeste** din fisier o valoare si o stocheaza in price[i]
10. inchide fisierul
end *function*

Am creat o funcție auxiliară **"maxim"** care returnează valoarea maximă dintre două numere pentru a ajuta la calculul venitului maxim obținut din vânzarea bucăților tijei.

Pseudocodul funcției

```
function maxim(a, b)
1.   daca a este mai mare ca b
2.        returneaza a
3.   returneaza b
```

```
end function
```

Am realizat funcția **"cutRod_v1"** pentru a determina cel mai mare venit obținut prin tăierea unei tije de lungime n si vânzarea porțiunilor rezulate cu o anumită valoare specificată in vectorul de prețuri. Această funcție folosește o abordare de programare dinamică și construiește treptat soluția. Vectorul **"partial"** stochează rezultatele intermediare pentru prețurile obținute din diferite combinații ale dimensiunilor.

Pseudocodul funcției

*func*tion **cutRod_v1(*p*rices[], n)**
1. intreg **i, j**
2. intreg **max_val**
3. initializeaza **"partial[0]"** cu 0
4. **pentru** i = 1, n executa
5. seteaza max_val la valoarea **INT_MIN**
6. **pentru** j = 0, i-1 executa
7. calculeaza **max_val** ca **maxim**(max_val, prices[j] + partial[i-j-1])
8. seteaza partial[i] la valoarea **max_val**
9. **returneaza** partial[n]

end *function*

A doua varianta pentru rezolvarea problemei este realizata prin functia recursiva **"cutRod_kp_v2"** care abordeaza metoda problemei rucsacului. Aceasta functie utilizeaza o matrice bidimensionala **"partial_price"** care stocheaza rezultatele intermediare pentru preturile obtinute din diferite combinatii ale dimensiunilor.

Pseudocodul funcției

*func*tion **cutRod_kp_v2(*p*rice[], length[], dim_max, dim)**

1. **dacă** dim =0 **atunci**
2. **returneaza** 0
3. **dacă** length[dim - 1] mai mic sau egal ca dim_max **atunci**
4. calculeaza **partial_price[dim][dim_max]** ca **maxim**(price[dim - 1] + **cutRod_kp_v2**(price, length, dim_max - length[dim - 1], dim), **cutRod_kp_v2**(price, length, dim_max, dim - 1))
5. **altfel**
6. calculeaza **partial_price[dim][dim_max]** ca **cutRod_kp_v2**(price, length, dim_max, dim - 1)
7. **returneaza** partial_price[dim][dim_max]

end *function*

Functia **"save_result"** este realizata pentru a salva rezultatele obtinute in urma rularii celor doi algoritmi asupra datelor din fisierele de teste.

Pseudocodul funcției

*func*tion **save_result(*r*esult1, result2)**
1. deschide fisierul **result.txt** in modul de adaugare
2. **dacă** nu s-a putut deschide fisierul **atunci**
3. **afișează** mesajul **"Fisierul nu poate fi deschis"**
4. se iese cu codul de eroare **"-1"**
5. scrie numele fisierului cu datele de intrare care se testeaza
6. scrie in fisier rezultatul obtinut in urma rularii primei functii (*"cutRod_v1"*)
7. scrie in fisier rezultatul obtinut in urma rularii celei de-a doua functii (*"cutRod_kp_v2"*)

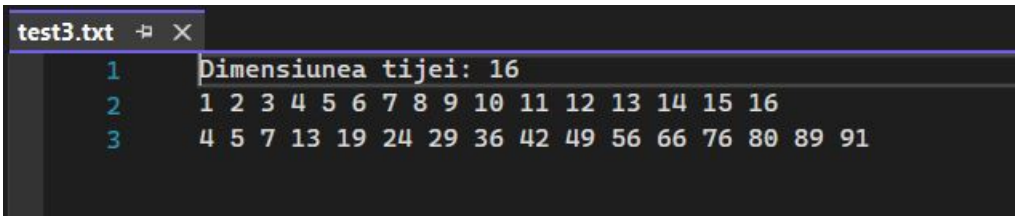
end *function*

3 Date experimentale și rezultate

3.1 Prezentarea datelor de intrare

Ca date experimentale am folosit datele generate in fiserele test1.txt ... test10.txt prin intermediul functiei create_test(), descrisa in capitolul anterior. Aceste au fost incarcate pentru a fi folosite cu ajutorul functiei load_test(), de asemenea descrisa in capitolul anterior. Datele sunt prelucrate prin functiile cutRod_v1 si cutRod_kp_v2 in vederea obtinerii venitului maxim obtinut din vanzarea diferitelor dimensiuni din tija.

Formatul unui test este urmatorul:



Fisierele cu datele care au fost testate au fost create astfel:

• pe prima linie este afisata dimensiunea tijei alaturi de un mesaj **“Dimensiunea tijei: “**

• pe a doua linie sunt afisate dimensiunile in care poate fi taiata tija

• pe a treia linie sunt afisate preturile pentru fiecare dimensiune a tijei.

Au fost create 10 seturi de date in care tija primeste dimensiuni diferite.Am considerat trei categorii de teste, in functie de aceste dimensiuni:

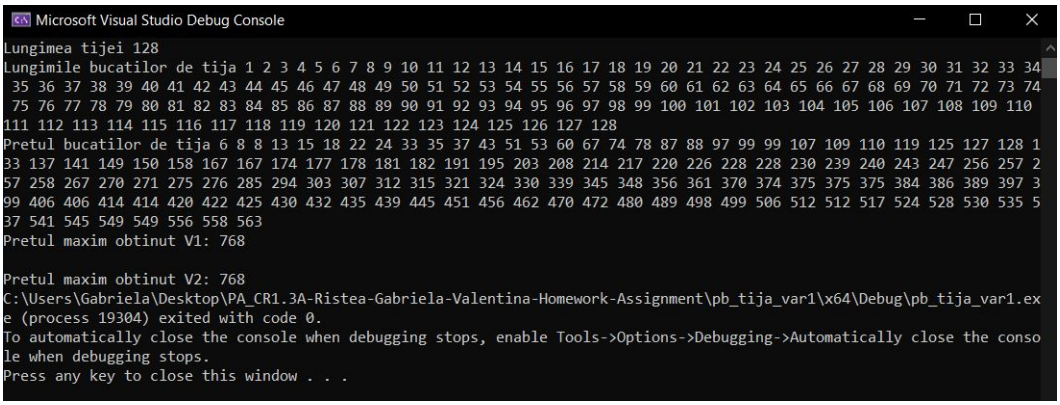
• teste mici - in care tija are dimensiuni in intervalul [4,20]

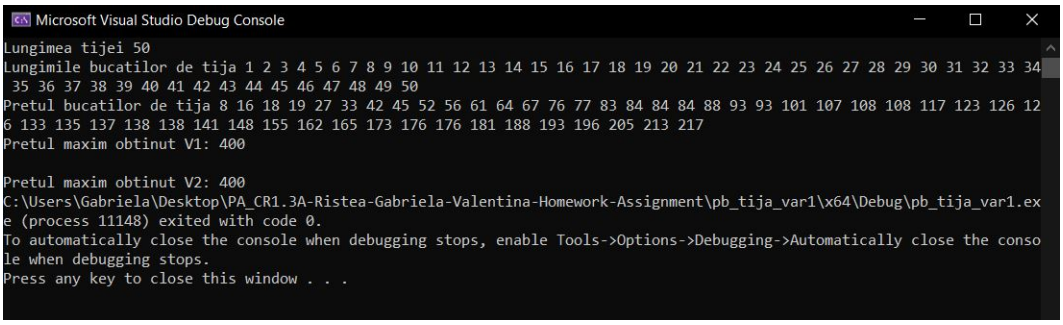
• teste medii - in care tija are dimensiuni in intervalul [21,80]

• teste mari - in care tija are dimensiuni in intervalul [81,128].

3.2 Prezentarea datelor de ieșire

Rezultatele sunt afisate astfel:





• pe prima linie este afisata dimensiunea tijei

• pe a doua lnie sunt afisate dimensiunile in care poate fi taiata tija

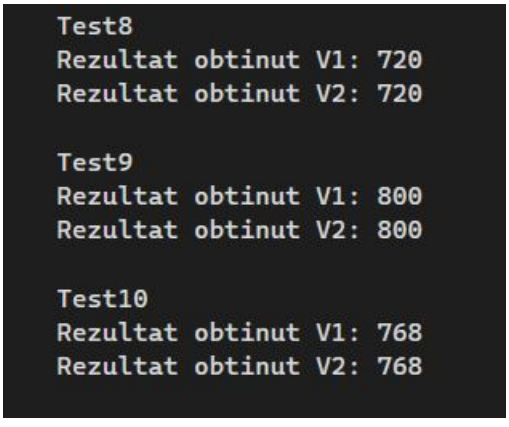
• pe urmtoarea linie sunt afisate preturile pentru fiecare dimensiune a tijei

• penultima linie contine venitul maxim care poate fi obtinut din vanzarea bucatilor de tija furnizat prin intermediul functiei *cutRod_v1 ()*

• ultima linie contine venitul maxim care poate fi obtinut din vanzarea bucatilor de tija furnizat prin intermediul functiei *cutRod_kp_v2 ()*.

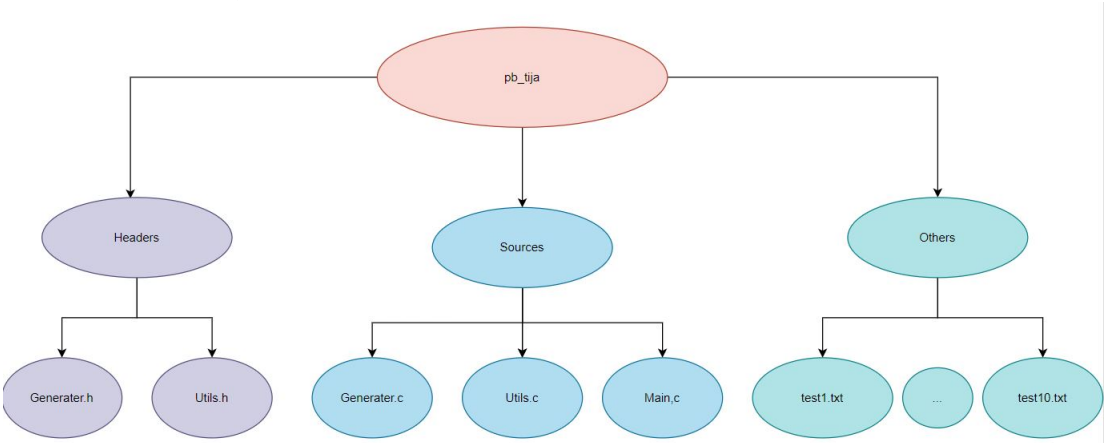
Rezultatele obtinute in urma rularii celor doi algoritmi asupra datelor din cele 10 teste sunt salvate in fisierul result.txt in urmatorul format:


result.txt
test10.txt
Test1
Rezultat obtinut V1: 22
Rezultat obtinut V2: 22
Test2
Rezultat obtinut V1: 64
Rezultat obtinut V2: 64
Test3
Rezultat obtinut V1: 93
Rezultat obtinut V2: 93
Test4
Rezultat obtinut V1: 200
Rezultat obtinut V2: 200
Test5
Rezultat obtinut V1: 155
Rezultat obtinut V2: 155
Test6
Rezultat obtinut V1: 400
Rezultat obtinut V2: 400
Test7
Rezultat obtinut V1: 367
Rezultat obtinut V2: 367


Test8
Rezultat obtinut V1: 720
Rezultat obtinut V2: 720
Test9
Rezultat obtinut V1: 800
Rezultat obtinut V2: 800
Test10
Rezultat obtinut V1: 768
Rezultat obtinut V2: 768

4 Proiectarea experimentală a aplicației

Structura aplicației:



Codul este modular avand trei fisere sursa (.c): **Generater.c, Utils.c, Main.c** si doua fisere headers (.h): **Generater.h si Utils.h**.

Fisierul header **Generater.h** contine definitiile functiilor *create_test()* pentru creare seturilor de date si *load_test()* pentru incarcarea acestora. Corpul acestor functii este scris in fisierul sursa **Generater.c**.

Fisierul header **Utils.h** continue definitiile functiilor auxiliare *maxim ()* si *save_result()* si a functiilor *cutRod_v1 ()* si *cutRod_kp_v2 ()*, care furnizeaza rezultatul problemei in doua variante descrise in capitolele anterioare. Corpul acestor functii este scris in fiserul sursa **Utils.c**.

În fișierul principal **main.c** se găsește funcția principală **int main()** prin intermediul căreia sunt generate întai testele (în interiorul ei este apelată functia de generare de teste aleatoare), după care se comentează această funcție, apoi se apelează cea de incarcare pentru obtinerea datelor care vor fi prelucrate prin intermediul apelurilor functiilor *cutRod_v1 ()* si *cutRod_kp_v2 ()*. Pentru salvarea rezultatelor obținute este apelată funcția *save_result ()*.

5 Complexitățile

Complexitatea in timp a functiei *cutRod_v1 ()* este O(n^2) deoarece exista doua bucle *for* inlantuite.

Complexitatea in spatiu a functiei *cutRod_v1 ()* este O(n) deoarece utilizeaza un vector numit *partial* de dimensiune n+1 pentru a stoca rezultatele partiale.

Complexitatea in timp a functiei *cutRod_kp_v2 ()* este O(n^2) deoarece depinde de numarul total de apeluri recursive.

Complexitatea in timp a functiei *create_test()* este O(n) deoarece necesita bucla *for* pentru generarea valorilor. Complexitatea in spatiu a acestei functii este, de asemenea, O(n).

Complexitatea in timp a functiei *load_test()* este O(n) deoarece necesita bulca *for* pentru incarcarea valorilor. Complexitatea in spatiu a acestei functii este, de asemenea, O(n).

6 Concluzii

Am luat aceasta tema ca pe o provocare pe care, dupa multe incercari, am reusit s-o duc la capat. Odata cu realizarea acestui proiect, am incercat sa ma familiarizei cu sintaxa din Latex.

Am ales sa folosesc ca mediu de dezvoltare VisualStudio Community 2022.

Am incercat sa implementez rezolvarea in doua variante conform cerintelor din metodologie.

Am implementat, de asemenea, functia necesara generarii de date de intrare non-triviale, precum si o functie care stocheaza rezultatele obtinute in urma prelucrarii acestor date.

In urma rularii am observat ca pentru o dimensiune a tijei mai mare functia *cutRod_kp_v2 ()* necesita un timp de executie mai indelungat. Pentru executarea testului 10, aceasta functie a furnizat rezultatul dupa aproximativ 30 de minute.

Cu toate acestea, ambele functii isi indeplinesc scopul si anume acela de a calculca venitul maxim obtinut in urma vanzarii portiunilor din tija.

7 Bibliografie

Bibliografie

[1] LaTeX project site
- https://www.overleaf.com/learn/how-to/Creating_a_document_in_Overleaf

[2] Knapsack Problem
- https://dyclassroom.com/dynamic-programming/0-1-knapsack-problem

[3] Generarea random a numerelor
- https://www.scaler.com/topics/random-number-generator-in-c/