**Assignment #3 (5%)**

Submission deadline: Sunday, **March 27, 2022** **(23:59)**

**NAME: Talha Abdullah Punjabi QUID: 201903446**

**Chapters:** This assignment covers Chapter 3 and Chapter 4 in the textbook

**Important Notes (must read):**

1. When submitting your work, you must use Blackboard, **NO other means like email submissions, are accepted.**
2. Assignments are to be solved **individually**.
3. A mark of zero (0) will be awarded for the whole assessment in which plagiarism was found to occur. Even if a single question is plagiarized**, the whole assignment will get zero** (0).
4. Submit your work as instructed below ***before*** the deadline**. No extension will be provided.**
5. Along with the MS Word submission file, you must submit separate Java files for the programs in **Questions 1 & 2**. Put all these files in a folder named **Assignment2\_QUID**. Compress this folder and submit it.
6. In the Word document, make sure that to add screenshots for input and output of your programs.

If you have any questions or doubts about any of the above-mentioned issues, please consult Eng. Alaa Hussein [alaa.hussein@qu.edu.qa](mailto:alaa.hussein@qu.edu.qa) . There are 5 questions in this assignment, each of them is 20 points.

**Q1. (Section 3.1.)**

Write **Java programs** for Algorithm 3.1. (Binomial Coefficient using Divide-and-Conquer) and Algorithm 3.2. (Binomial Coefficient using Dynamic Programming) and evaluate their performances using different problem instances. Explain what happens when k increases.

*[Grading policy: each implementation is 7.5 points and performance evaluation is 5 points]*

Answer: (please write your answer here, add required space if needed)

Ans)

Algo 3.1

**package** Assignment3;

**public** **class** Algorithm3\_1 {

**public** **static** **void** main(String [] args) {

//int n=7; int k=2;

**int** n=100; **int** k=5;

**long** start=System.*nanoTime*();

**int** answer=*binomialCoefficient*(n,k);

**long** end=System.*nanoTime*();

System.***out***.println("n=7,k=2, Answer :"+answer);

**long** performance=end-start;

System.***out***.println("Time taken by this algorithm is : "+performance);

}

**private** **static** **int** binomialCoefficient(**int** n, **int** k) {

// **TODO** Auto-generated method stub

**if**(k>n)

**return** 0;

**if** (k==0 || k==n)

**return** 1;

**return** *binomialCoefficient*(n-1,k-1)+*binomialCoefficient*(n-1,k);

}

}

Algo 3.2

**package** Assignment3;

**public** **class** Algorithm3\_2 {

**public** **static** **void** main(String [] args) {

//int n=7; int k=2;

**int** n=100; **int** k=5;

**long** start=System.*nanoTime*();

**int** ans=*binomialCoefficient*(n,k);

**long** end=System.*nanoTime*();

System.***out***.println("Answer is : "+ans);

**long** performance=end-start;

System.***out***.println("Time taken by this algorithm is : "+performance);

}

**private** **static** **int** binomialCoefficient(**int** n, **int** k) {

// **TODO** Auto-generated method stub

**int** A[][]=**new** **int**[n+1][k+1];

**int** i,j;

**for**(i=0;i<=n;i++)

**for**(j=0;j<=Math.*min*(i,k);j++)

{

**if**(j==0 || j==i)

A[i][j]=1;

**else**

A[i][j]=A[i-1][j-1]+A[i-1][j];

}

**return** A[n][k];

}

**When n and k are small**

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Algorithm 3.1(Divide and Conquer) is more efficient than Algorithm 3.2(Dynamic Programming)

**When n and k are big**

Text

Description automatically generated

Text

Description automatically generated

Algorithm 3.1(Divide and Conquer) is less efficient than Algorithm 3.2(Dynamic Programming)

**Performance Analysis**

Using divide and conquer can sometimes be efficient when n and k are less.

While, when using dynamic programming, after a value of n, this algorithm will be more efficient.

**Q2. (Section 4.1.)**

Write a **Java program** that implements Prim’s algorithm (Algorithm 4.1.) and evaluate its performance using different graphs.

*[Grading policy: implementation is 15 points and performance evaluation is 5 points]*

Answer: (please write your answer here, add required space if needed)

**package** Assignment3;

**public** **class** PrimsAlgo {

**public** **static** **void** main(String []args) {

**int** graph[][]=**new** **int**[][]{ {0,3,0,3,0},

{4,0,4,5,3},

{2,4,0,4,0},

{5,0,3,4,1},

{2,0,0,4,1}};

*prim*(graph);

}

**private** **static** **final** **int** ***A***=5;

**public** **static** **int** mkey(**int** key[], **boolean** msSet[])

{

**int** min=Integer.***MAX\_VALUE***, min\_index=-1;

**for** (**int** i=0;i<***A***;i++) {

**if**(msSet[i]==**false** && key[i]<min) {

min=key[i];

min\_index=i;

}

}

**return** min\_index;

}

**public** **static** **void** prim(**int** graph[][]) {

**int** p[]= **new** **int**[***A***];

**int** k[]= **new** **int**[***A***];

**boolean** msSet[]=**new** **boolean**[***A***];

**for** (**int** j=0;j<***A***;j++) {

k[j]=Integer.***MAX\_VALUE***;

msSet[j]=**false**;

}

k[0]=0;

p[0]=-1;

**for**(**int** c=0;c<***A***-1;c++) {

**int** u=*mkey*(k,msSet);

msSet[u]=**true**;

**for**(**int** a=0;a<***A***;a++) {

**if**(graph[u][a]!=0 && msSet[a]==**false** && graph[u][a]<k[a]) {

p[a]=u;

k[a]=graph[u][a];

}

}

}

*printPrim*(p,graph);

}

**public** **static** **void** printPrim(**int** p[], **int** g[][]) {

System.***out***.println("Edge Weight");

**for**(**int** i=1;i<***A***;i++) {

System.***out***.println(p[i]+" to "+i+" : "+g[i][p[i]]);

}

}

}

**Output:**

Table

Description automatically generated with medium confidence

**Q3. (Section 3.4.)**

Find the optimal order, and its cost, for evaluating the product A1 **X** A2 **X** A3 **X** A4 **X** A5, where

A1 is (10 X 4)

A2 is (4 X 5)

A3 is (5 X 20)

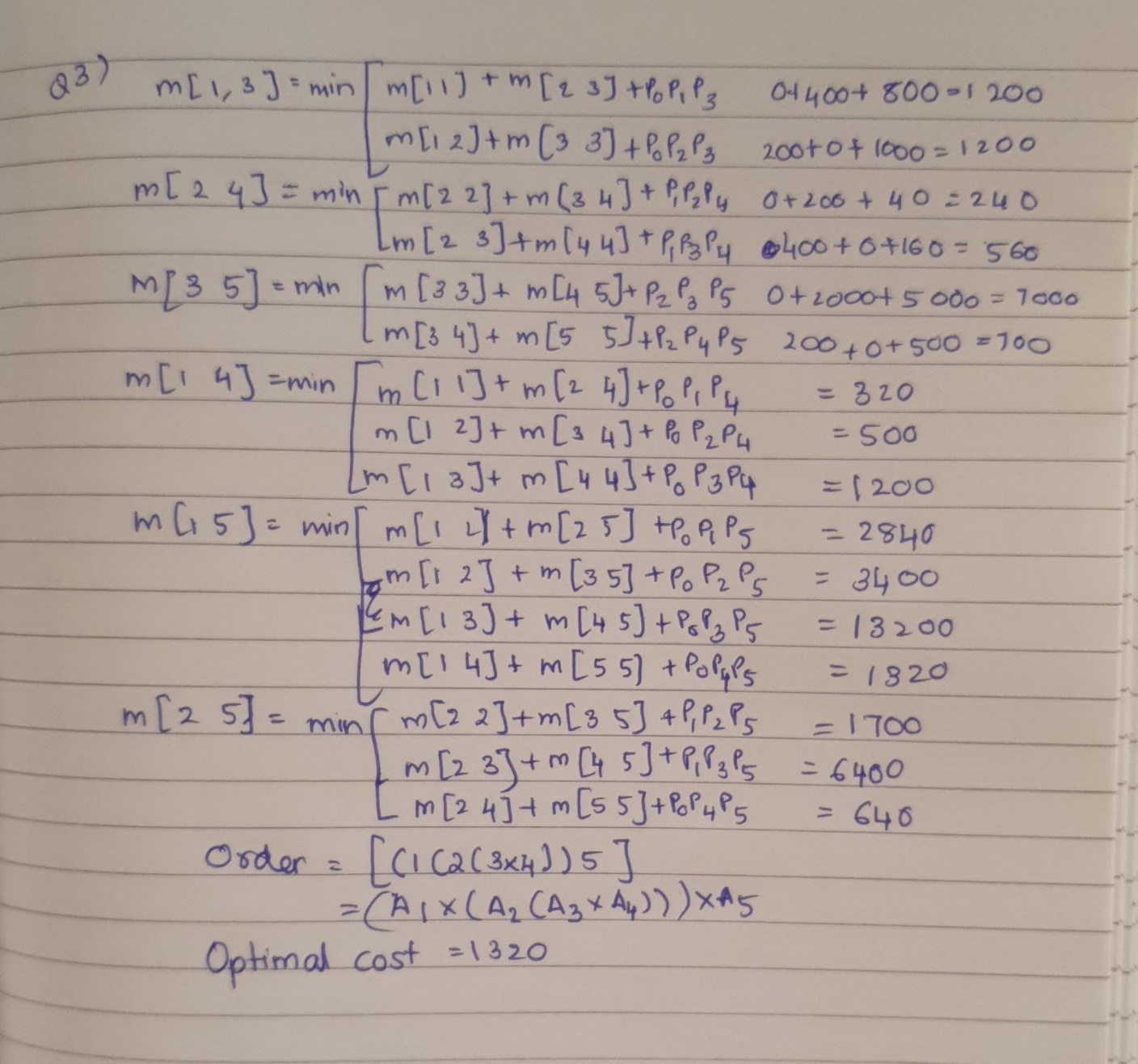
A4 is (20 X 2)

A5 is (2X50)

*[Grading policy: optimal order is 10 points and cost is 10 points]*

Answer: (please write your answer here, add required space if needed)

M[I,j] = m[I,k] +m[k+1,j] +Pi-1PkPj



**Q4. (Section 4.3.)**

Consider the following table that shows the jobs, deadlines, and profits. Use the Scheduling with Deadlines algorithm (Algorithm 4.4.) to maximize the total profit. Show how J and K arrays change in each step and specify the final J’s content.

|  |  |  |
| --- | --- | --- |
| **Job** | **Deadline** | **Profit** |
| **1** | **2** | **40** |
| **2** | **4** | **15** |
| **3** | **3** | **60** |
| **4** | **2** | **20** |
| **5** | **3** | **10** |
| **6** | **1** | **45** |
| **7** | **1** | **55** |

*[Grading policy: each correct step is 1.5 points, and the final correct J array is 4.5 points]*

Answer: (please write your answer here, add required space if needed)

Sorting based on job’s deadline and then profit

|  |  |  |
| --- | --- | --- |
| **Job** | **Deadline** | **Profit** |
| **7** | **1** | **55** |
| **6** | **1** | **45** |
| **1** | **2** | **40** |
| **4** | **2** | **20** |
| **3** | **3** | **60** |
| **5** | **3** | **45** |
| **2** | **4** | **15** |

Considering 6th job as number 2

Choosing job at least one less than previous deadline and highest profit

Taking deadline 4

Profit= 15

Now coming to the deadline 3

Profit of job 3 is more than job 5

Hence, profit = 15+60 =75

Coming to the deadline 2,

Profit of Job 1 is greater than Job 4

Profit= 75+40 = 115

Coming to the deadline 1

Profit of Job 7 is greater than Job 6

Profit = 115+55= 170 [ Profit is maximum as 170 ]

**Q5. (Section 4.4.)**

Use Huffman’s algorithm to construct an optimal binary prefix code for the letters in the following table.

**Letter:**  A B I M S X Z

**Frequency:** 12 7 18 10 9 5 2

*[Grading policy: Each correct prefix code is 3 points except the last character (2 points)]*

Answer: (please write your answer here, add required space if needed)

**63**

**26 37**

**A(12) 14 I(18) 19**

**B(7) 7 S(9) M(10)**

**X(5) Z(2)**