# Xi’an JIAOTONG-LIVERPOOL UNIVERSITY

西 交 利 物 浦 大 学

# Coursework Submission Cover Sheet

|  |  |  |
| --- | --- | --- |
| Name | Jin(Surname) | Minhao(Other Names) |
| Student Number | 1717576 | |
| Programme | Information and Computing Science | |
| Module Title | Algorithmic Foundations and Problem Solving | |
| Module Code | CSE102 | |
| Assignment Title | Assignment 1 | |
| Submission Deadline | 10-April- 2019, Wednesday 17:30 | |
| Module Leader | Dr. Muhammad Alam | |

By uploading or submitting this coursework submission cover sheet, I certify the following:

* I have read and understood the definitions of PLAGIARISM, COLLUSION, and the FABRICATION Of DATA, as outlined in the Undergraduate Student Handbook of Xi’an Jiaotong-Liverpool University and as posted on the University Website.
* This work is my own, original work produced specifically for this assignment. It does not misrepresent the work of another person or institution as my own. Additionally, it is a submission that has not been previously published, or submitted to another module.
* This work is not the product of unauthorized collaboration between myself and others.
* This work is free of embellished or fabricated data.

I understand that PLAGIARISM, COLLUSION and the FABRICATION OF DATA are serious disciplinary offences. By uploading or submitting this cover sheet, I acknowledge that I am subject to disciplinary action if I am found to have committed such acts.

Signature ……*.*…………………………………… Date ……*………………………………*……………

|  |  |  |  |
| --- | --- | --- | --- |
| For Academic Office use: | Date Received | Days Late | Penalty |
|  |  |  |

## Feedback on the strength of the work

Feedback on the weakness that needs to be improved

**1st Marker Date Mark 2nd Marker Date Mark**

**(if applicable)**

***Students:*** *Please start your assignment on the next page.*

**Module: CSE102 Assignment 1**

**Assessment**

The tasks contribute 10% to the overall assessment of CSE102

**Submission**

Please complete the assessment tasks using Microsoft Word and submit via ICE. You are also asked to print out a copy of your answersand submit it to my mail box at Block SD 4th Floor by 10-April- 2019, Wednesday.

**Deadline**

10-April- 2019, Wednesday 17:30.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 1**

Consider the following graph G.

*d*

*b*

*c*

*a*

*f*

*e*

1. Give the adjacency matrix and adjacency list of the graph *G*. **(5 marks)**

**Answer:**

* the adjacency matrix of the graph *G*:



* the adjacency list of the graph *G*:













1. Let *e1=(a, b), e2= (a, e), e3= (b, c) e4= (b, f), e5= (c, f), e6= (c, d), e7= (e, f),* give the incidence matrix and incidence list of the graph *G*. **(5 marks)**

**Answer:**

* the incidence matrix of the graph *G*:



* the incidence list of the graph *G*:















1. Starting at the vertex ***a*** and resolving ties by the vertex alphabetical order traverse the graph by breadth-first-search (BFS) and construct the corresponding BFS tree. **(5 marks)**

**Answer:**

The travel order is a, b, e, c, f, d.

The BFS tree is depicted below.



1. Starting at the vertex ***a***and resolving ties by the vertex alphabetical order traverse the graph by depth-first-search (DFS) and construct the corresponding DFS tree. **(5 marks)**

**Answer:**

The travel order is a, b, c, d, f, e.

The BFS tree is depicted below.



**Question 2**

Consider the following recursive function

*f*(n)=

1. Write a recursive (top-down) algorithm to compute it. (**5 marks)**

**Answer:**

Procedure *f*(n)

if n==0 or n==1or n==2 or n==3 then

return 1

else

return *f* (n-1) + *f* (n-2) + *f* (n-3)

1. Design and write the pseudo code of a faster nonrecursive (bottom-up) algorithm using the concept of dynamic programming.**(5marks)**

**Answer:**

Procedure *f*(n)

Set A[0] = A[1] = A[2] = A[3]=1

for i = 4 to n do

A[i] = A[i-1] + A[i-2] + A[i-3]

return A[n]

**Question 3**

Suppose there are three assembly lines each with 4 stations, *Si,j* (i=1, 2, 3; j=1, 2, 3, 4). The assembly time is given in the circle representing the station and the transfer time is given next to the edge from one station to another.

*S1,2*

*S1,3*

1

1

1

1

1

1

1

1

1

2

3

1

1

0

1

1

2

4

1

0

0

0

0

0

finish

start

3

3

4

8

*S2,1*

*S2,2*

*S2,3*

*S2,4*

0

0

0

3

7

8

2

*S3,1*

*S32*

*S33*

*S34*

4

3

6

3

*S1,1*

*S1,4*

1. Using dynamic programming, fill in the following table of the minimum time *fi*[*j*] needed to get through station *Si,j* and the *line* of the station just before *Si,j* on the fastest way to get through *Si,j*. Show all the intermediate steps in computing these values. **(15 marks)**

**Answer:**

|  |  |  |  |
| --- | --- | --- | --- |
| *j* | *f1*[*j*] | *f2*[*j*] | *f3*[*j*] |
| 1 | 3 | 4 | 3 |
| 2 | 6 | 7 | 10 |
| 3 | 10 | 13 | 15 |
| 4 | 18 | 14 | 13 |

* *f1*[*1*]=3

*f2*[*1*]=4

*f3*[*1*]=3

* *f1*[*2*]=min{ *f1*[*1*]+0+3, *f2*[*1*]+1+3, *f3*[*1*]+3+3}=6

*f2*[*2*]=min{ *f1*[*1*]+1+3, *f2*[*1*]+0+3, *f3*[*1*]+2+3}=7

*f3*[*2*]=min{ *f1*[*1*]+1+7, *f2*[*1*]+1+7, *f3*[*1*]+0+7}=10

* *f1*[*3*]=min{ *f1*[*2*]+0+4, *f2*[*2*]+2+4, *f3*[*2*]+1+4}=10

*f2*[*3*]=min{ *f1*[*2*]+4+6, *f2*[*2*]+0+6, *f3*[*2*]+1+6}=13

*f3*[*3*]=min{ *f1*[*2*]+1+8, *f2*[*2*]+1+8, *f3*[*2*]+0+8}=15

* *f1*[*4*]=min{ *f1*[*3*]+0+8, *f2*[*3*]+1+8, *f3*[*3*]+1+8}=18

*f2*[*4*]=min{ *f1*[*3*]+1+3, *f2*[*3*]+0+3, *f3*[*3*]+1+3}=14

*f3*[*4*]=min{ *f1*[*3*]+1+2, *f2*[*3*]+1+2, *f3*[*3*]+0+2}=13

1. What is the minimum time *f*\* needed to get through the assembly line? **(5 marks)**

**Answer:**

*f*\*=min{ *f1*[*4*], *f2*[*4*], *f3*[*4*]}=13

1. Based on the line information on the table, show how to find the fastest way (which stations should be chosen?) **(10 marks)**

**Explanation**

f\*= *f*3[*4*], the last station is S34; backward from *f*3[*4*], as *f*3[*4*]= *f1*[3]+1+2, the station before S34 is S13; backward from *f*1[*3*], as *f*1[*3*]= *f1*[2] + 0 + 4, the station before S13 is S12; backward from *f*1[*2*], as *f*1[*2*]= *f1*[1] + 0+3, the station before S12 is S11. Finally we can find the following fastest way.

**S11 -> S12-> S13->S34**

**Question 4**

Consider the following graph G. The label of an edge is the cost of the edge.

8

1

6

4

10

10

3

1

*a*

*b*

*c*

*d*

*e*

*f*

*g*

2

5

1. Using *Prim's* algorithm, draw a *minimum spanning tree* (MST) of the graph Also write down the change of the priority queue step by step and the order in which the vertices are selected. Is the MST drawn unique? (i.e., is it the one and only MST for the graph?) **[5 marks]**

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vertex order | a(0,-) | b(-,∞) | c(-,∞) | d(-,∞) | e(-,∞) | f(-,∞) | g(-,∞) |
| a(0,-) |  | b(-,∞) | c(-,∞) | d(-,∞) | e(a,1) | f(a,4) | g(-,∞) |
| e(a,1) |  | b(-,∞) | c(-,∞) | d(-,∞) |  | f(e,2) | g(-,∞) |
| f(e,2) |  | b(f,3) | c(f,10) | d(-,∞) |  |  | g(f,5) |
| b(f,3) |  |  | c(b,6) | d(b,8) |  |  | g(f,5) |
| g(f,5) |  |  | c(g,1) | d(b,8) |  |  |  |
| c(g,1) |  |  |  | d(b,8) |  |  |  |
| d(b,8) |  |  |  |  |  |  |  |



The MST is unique.

1. Using *Kruskal’s* algorithm, draw a *minimum spanning tree* (MST) of the graph G. Write down the order in which the edges are selected.

Is the MST drawn unique? (i.e., is it the one and only MST for the graph?)

**(5 marks)**

**Answer:**

|  |  |
| --- | --- |
| **(a, e)** | **1✓** |
| **(c, g)** | **1✓** |
| **(e, f)** | **2✓** |
| **(b, f)** | **3✓** |
| **(a, f)** | **4🗶** |
| **(f, g)** | **5✓** |
| **(b, c)** | **6🗶** |
| **(b, d)** | **8✓** |
| **(c, f)** | **10🗶** |
| **(d, f)** | **10🗶** |



The MST is unique.

1. Referring to the same graph above, find the shortest paths from the vertex ***a*** to *all* other vertices in the graph G using *Dijkstra’s* algorithm. Show the changes of the priority queue step by step and give the order in which edges are selected. **(10 marks)**

N.B. There may be more than one solution. You only need to give one of the solutions.

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vertex order | a(0,-) | b(-,∞) | c(-,∞) | d(-,∞) | e(-,∞) | f(-,∞) | g(-,∞) |
| a(0,-) |  | b(-,∞) | c(-,∞) | d(-,∞) | e(a,1) | f(a,4) | g(-,∞) |
| e(a,1) |  | b(-,∞) | c(-,∞) | d(-,∞) |  | f(e,3) | g(-,∞) |
| f(e,3) |  | b(f,6) | c(f,13) | d(-,∞) |  |  | g(f,8) |
| b(f,6) |  |  | c(b,12) | d(b,14) |  |  | g(f,8) |
| g(f,8) |  |  | c(g,9) | d(b,14) |  |  |  |
| c(g,9) |  |  |  | d(b,14) |  |  |  |
| d(b,14) |  |  |  |  |  |  |  |

**Question 5**

Let A[0..n−1] be an array of n real numbers. A pair (A[i], A[j]) is said to be an inversion if these numbers are out of order, i.e. i < j but A[i] > A[j]. Design an O(n log n) algorithm for counting the number of inversions. (**20 marks**).

**Answer:**

**algorithm Mergesort(A[0..n-1], int start, int end)**

**Set inversionCount = 0;**

**Set length = end - start;**

**if length > 1 then**

**begin**

**Set mid = (start + end) / 2**

**inversionCount += mergeSort(A, start, mid)**

**inversionCount += mergeSort(A, mid, end)**

**inversionCount += merge(arr, start, mid, end)**

**end**

**return inversionCount**

**algorithm merge(int[] arr, int start, int mid, int end)**

**if arr == null || start < 0 || end > arr.length then**

**return 0**

**int[] temp = new int[end - start]**

**Set inversionCount = 0;**

**Set i = start**

**Set j = mid**

**Set k = 0**

**while i < mid and j < end do**

**begin**

**if arr[i] <= arr[j] then set temp[k] = arr[i] and increase i**

**else set temp[k] = arr[j] and increase j**

**k=k+1**

**inversionCount += mid - i;**

**end**

**if i != mid then copy arr[i…mid] to temp[k…k+mid-i]**

**if j != end then copy arr[j…end] to temp[k…k+end-j]**

**copy temp[0…temp.length-1] to arr[start… start+ temp.length-1]**

**return inversionCount;**