CS 260 Homework 6

Semanti Basu

August 11, 2017

Question 1

Adjacency matrix giving arc costs

*	a	b	c	d	e	f
a	0	3	∞	4	∞	5
b	∞	0	1	∞	∞	1
c	∞	∞	0	2	∞	∞
d	∞	3	∞	0	∞	∞
e	∞	∞	∞	3	0	2
f	∞	∞	∞	2	∞	0

Linked Adjacency List with arc costs

Node	Connects To
a→	(b,3),(d,4),(f,5)
b→	(c,1),(f,1)
c→	(d,2)
d→	(b,3)
e→	(f,2),(d,3)
f→	(d,2)

Question 2

The tasks T1,T2 etc can be the nodes on the graph. The required times can be encoded on the edges. The graph will be directed graph. If Ti should be completed prior to Tj, the edge should be directed from node Ti to Tj.

If no parallel execution occurs, then the time taken= t1+t2+t3+...+tn

Else, the shortest path in the graph containing all the nodes will have to be found and the values on the edges of the graph added.For parallel execution critical path length will have to be found.

If parallel execution allowed, then minimum time=MAX(t1,t2,t3...tn). In this graph, all nodes will branch from or be adjacent to one particular node.

Since, parallel execution, we have to wait for the longest running process to finish.

The DFS algorithm visits every node and every edge. So, it can be used to calculate minimum time.O(n*tn) is the running time.

Question 3

1. b→ c→ d
2. b→ f→ d
3. c→ d→ b→ f

Question 4

Insert edges

Adjacency list uses linked list to represent all the adjacent nodes of a particular node.Let the linked list l[i] represent all the nodes adjacent to node i. Let l[j] represent the list of all nodes adjacent to node j. To insert an edge between nodes i and j of an undirected graph, node i has to be added to the linked list for node j. Similarly node j has to be added to the adjacency list for node i.

To do so, the linked list for node i containing all the nodes of the graph adjacent to node i will have to be accessed. This list has to be traversed till the end, and then node j will have to be inserted in the list. Similarly the linked list for node j containing all the nodes of the graph adjacent to node j will have to be accessed. This list has to be traversed till the end, and then node i will have to be inserted in the list.

Let there be n nodes in graph.The adjacency list representation can be considered as an array of linked lists.

Data: nodes i and j
Result: Insert an edge between nodes i and j
if *i>n or i<0* **then**
| return Invalid node;
end
if *j>n or j<0* **then**
| return Invalid node;
end
a=l[i]→head;
for *k=0,i ≤ length(l[i])-1,k++* **do**
| a=a→next;
end
a→next=j;
b=l[j]→head;
for *k=0,i ≤ length(l[j])-1,k++* **do**
| b=b→next;
end
b→next=i;

Delete edges

To delete an edge between nodes i and j, the algorithm should be designed to search the linked list l[i] to lookup node j and delete it. Similarly l[j] should be searched for node i and i should be removed from l[j].

```
Data: nodes i and j
Result: Delete an edge between nodes i and j
if i>n or i<0 then
    return Invalid node;
end
if j>n or j<0 then
    return Invalid node;
end
a=l[i]→head;
for k=0,i ≤ length(l[i])-1,k++ do
    if a==j then
        delete j;
        break;
    end
    a=a→next;
end
if a→next==NULL then
    print Edge doesnt exist;
end
b=l[j]→head;
for k=0,i ≤ length(l[j])-1,k++ do
    if b==j then
        delete j;
        break;
    end
    b=b→next;
end
if b→next==NULL then
    print Edge doesnt exist;
end
```