COMP3711: Design and Analysis of Algorithms

Tutorial 10

HKUST

Question 1

Let G be a connected undirected graph with distinct weights on the edges, and let e be an edge of G. Suppose e is the largest-weight edge in some cycle of G. Show that e cannot be in the MST of G.

Solution 1

Let T be the MST of G and suppose T contains $e = (u, v)$. Removing e from T breaks it into S and $V - S$. Consider the cycle that has e as the largest-weight edge. This cycle starts from u, goes to v, and takes another path to go back to u. This path must cross this cut somewhere via an edge e' with $w(e') < w(e)$. Now we add e' to T and this turns it back to a spanning tree T'. We see that T' has a smaller weight than T, which contradicts with the early assumption that T is the MST. Thus T cannot contain e.

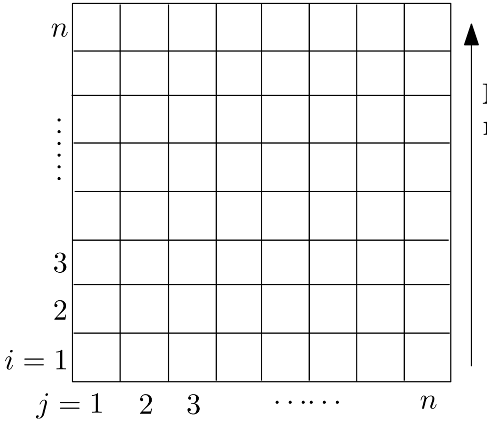
Question 2

Suppose that you are given an $n \times n$ checkerboard and a checker. You must move the checker from the bottom edge of the board to the top edge of the board according to the following rule. At each step you may move the checker to one of three squares:

1) the square immediately above,
2) the square that is one up and one to the left (but only if the checker is not already in the leftmost column),
3) the square that is one up and one to the right (but only if the checker is not already in the rightmost column).

Question 2

Checkerboard



Move the checker from row 1 to row n

Question 2

Each time you move from square (i, j) to square (i', j'), you receive $p((i, j), (i', j'))$ dollars. You are given a list of the values $p((i, j), (i', j'))$ for each pair $((i, j), (i', j'))$ for which a move from (i,j) to (i',j') is legal. Do not assume that $p((i, j), (i', j'))$ is positive.

Give an algorithm that figures out the set of moves that will move the checker from somewhere along the bottom edge to somewhere along the top edge while gathering as many dollars as possible. You algorithm is free to pick any square along the bottom edge as a starting point and any square along the top edge as a destination in order to maximize the number of dollars gathered along the way. What is the running time of your algorithm?

Solution 2

Denote each square by the pair (i, j), where i is the row number, j is the column number, and $1 \leq i, j \leq n$. Our goal is to find a most profitable way from any square in row 1 to any square in row n. Once we do so, we can look up all the most profitable ways to get to any square in row n and pick the best one.

A subproblem is the most profitable way to get from some square in row 1 to a particular square (i, j). We have optimal substructure as follows. Consider a subproblem for (i, j), where $i > 1$, and consider the most profitable way to (i, j).

Because of how we define legal moves, it must be through square $(i-1, j')$, where $j' = j-1, j$, or $j+1$. Then the way that we got to $(i-1, j')$ within the most profitable way to (i, j) must itself be a most profitable way to $(i-1, j')$. Thus we have the following space of subproblems and the recurrence:

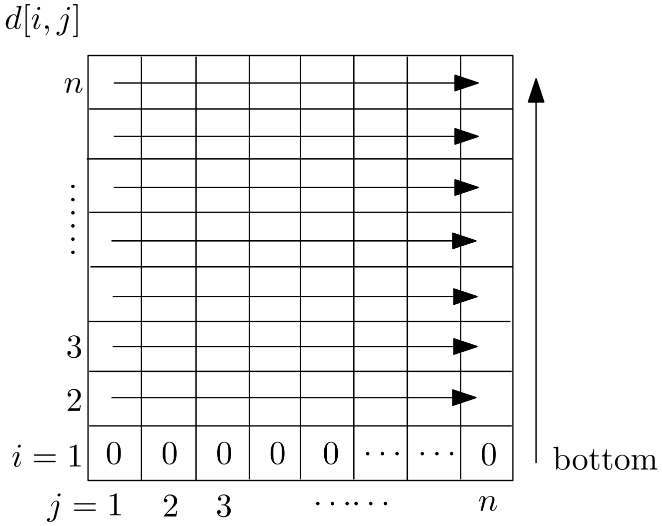
Solution 2

Let $d[i,j]$ be the profit we earn in the most profitable way to (i,j). Then we have that $d[1,j] = 0$ for all $j = 1, 2, ..., n$. For $i = 2, 3, ..., n$, we have

$$d[i,j] = \max \begin{cases} d[i-1,j-1] + p((i-1,j-1),(i,j)) & \text{if } j > 1, \\ d[i-1,j] + p((i-1,j),(i,j)) & \text{always}, \\ d[i-1,j+1] + p((i-1,j+1),(i,j)) & \text{if } j < n. \end{cases}$$

We can compute all the $d[i,j]$ row by row from $i = 1$ to $i = n$. Once we fill in the $d[i,j]$ table, the profit earned by the most profitable way to any square along the top row is $\max_{1 \le j \le n}\{d[n,j]\}$.
To keep track of how we got to (i,j) most profitably, we let $w[i,j]$ be the value of j' used to achieve the maximum value of $d[i,j]$. These values are defined for $2 \le i \le n$ and $1 \le j \le n$.

Solution 2



$d[i, j]$
n
$\vdots$
3
2
$i = 1$
0 0 0 0 0 $\cdots$ $\cdots$ 0
$j = 1$ 2 3 $\cdots\cdots$ n
bottom

Solution 2

```
for $j = 1$ to $n$ do
    $d[1, j] = 0$;
end
for $i = 2$ to $n$ do
    for $j = 1$ to $n$ do
        $up = d[i-1, j] + p((i-1, j), (i, j))$;
        if $(j = 1)$ then
            $upLeft = -\infty$;
            $upRight = d[i-1, j+1] + p((i-1, j+1), (i, j))$;
        else if $(j = n)$ then
            $upLeft = d[i-1, j-1] + p((i-1, j-1), (i, j))$;
            $upRight = -\infty$;
        else
            $upLeft = d[i-1, j-1] + p((i-1, j-1), (i, j))$;
            $upRight = d[i-1, j+1] + p((i-1, j+1), (i, j))$;
        end
        d[i,j] = max($upLeft, up, upRight$);
        if $(d[i, j] = upLeft)$ then
            $w[i, j] = j - 1$;
        else if $(d[i, j] = up)$ then
            $w[i, j] = j$;
        else
            $w[i, j] = j + 1$;
        end
    end
end
```

Solution 2

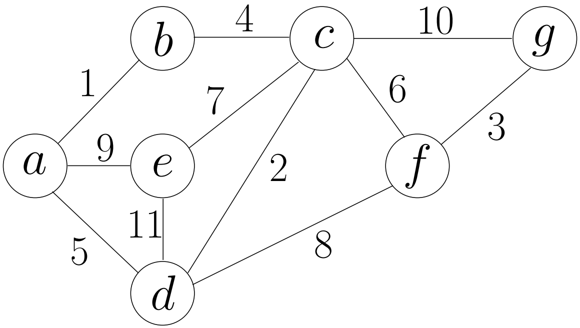
To actually compute the set of moves, we use the usual recursive backtracking method. This procedure prints the squares visited, from row 1 to row n:

```
PRINT-MOVES($w, i, j$)
**if** $i > 1$
    PRINT-MOVES($w, i-1, w[i, j]$)
print "(" $i$ "," $j$ ")"
```

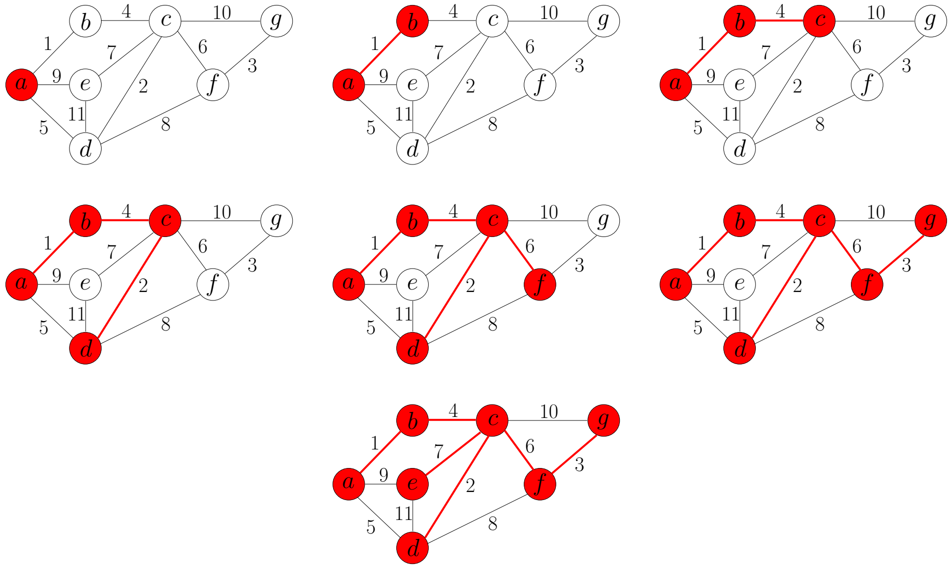
Letting $t = \max_{1 \leq j \leq n}\{d[n, j]\}$, the initial call is PRINT-MOVES(w, n, t).
The time to fill the $d[i, j]$ and $w[i, j]$ tables is clearly $O(n^2)$. Once we have computed the d and w tables, PRINT-MOVES runs in $O(n)$ time.

Question 3

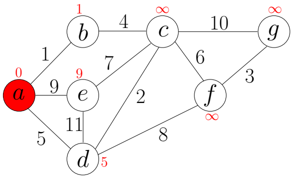
Prim's minimum spanning tree algorithm and Dijkstra's shortest path algorithm are very similar, but with crucial differences. Run both algorithms on the following graph, and show the partial MST / shortest path tree after every new edge has been added. The starting vertex for both algorithms is "a".

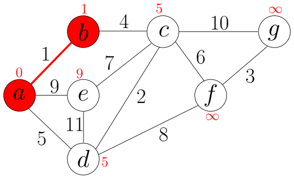


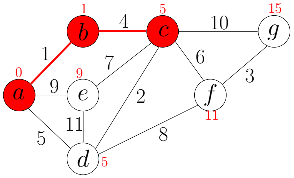
Solution 3 (MST)

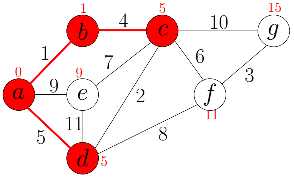


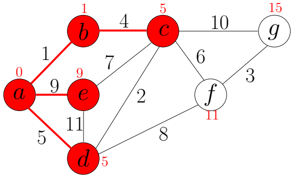
Solution 3 (Shortest path tree)

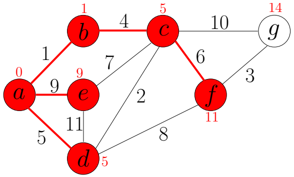

a b c d e f g
0 1 ∞ 5 9 ∞ ∞
1 4 10 9 7 6 2 3 11 5 8

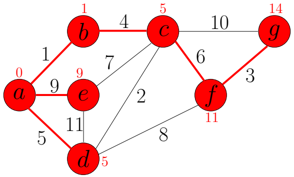

a b c d e f g
0 1 5 5 9 ∞ ∞
1 4 10 9 7 6 2 3 11 5 8


a b c d e f g
0 1 5 5 9 11 15
1 4 10 9 7 6 2 3 11 5 8


a b c d e f g
0 1 5 5 9 11 15
1 4 10 9 7 6 2 3 11 5 8


a b c d e f g
0 1 5 5 9 11 15
1 4 10 9 7 6 2 3 11 5 8


a b c d e f g
0 1 5 5 9 11 14
1 4 10 9 7 6 2 3 11 5 8


a b c d e f g
0 1 5 5 9 11 14
1 4 10 9 7 6 2 3 11 5 8