

Theory Homework 4 Solution

Problem 1 (25pts)

Solution

```
partition(lo, hi, A)
    // base case
    if lo == (hi + 1), done

    leftPtr = lo
    leftEqual = lo
    rightPtr = hi
    rightEqual = hi
    Determine random pivot, p

    while(leftPtr != rightPtr)

        while(A[leftPtr++] <= p)
            if A[leftPtr] == p
                swap A[leftEqual], A[leftPtr]
                leftEqual++

        while(A[rightPtr--] >= p)
            if A[rightPtr] == p
                swap A[rightEqual], A[rightPtr]
                rightEqual--

        swap A[leftPtr], A[rightPtr]

    swap left at leftPtr from A[lo] to A[leftEqual]
    swap right at rightPtr from A[rightEqual] to A[hi]
```

Problem 2 (25pts)

Solution

(a) Project all the sticks to the xy plane. If the projections of two sticks do not intersect, we know they are unrelated; otherwise, suppose the intersection point of them is (p, q) , then we can compute the z -coordinate of each stick by substituting the x, y value of the line equation corresponding to each stick using p , and q respectively, and the stick with a larger z -coordinate value is above the other.

Pseudocode:

```
// returns the spatial relation between sticks a, b
SpatialRelationSticks(a,b) {

    If(projections of a and b to x-y plane do not intersect)
    // get x-y slopes of sticks
    // find point of intersection between sticks=(p,q)
    // sticks intersect if point of intersection is within the bounds of
    // both sticks

        Return unrelated;

    else {
        compute the z - coordinate of each stick (denoted as z(a) and z(b)) by
        substituting the x, y value of the line equation
        corresponding to each stick using p and q respectively;

        if(z(a)>z(b))
            Return above;
        Else
            Return below;
    }
}
```

(b) Construct a directed graph $G = (V, E)$ as follows. Each vertex of V corresponds to one stick. And a directed edge is constructed from vertex a to vertex b , if the corresponding sticks, denoted as $s(a)$ and $s(b)$, satisfy $s(a)$ is above $s(b)$. Then we can run Topological Sort algorithm on G . If there is cycle in G found by Topological Sort algorithm, we know that it is not possible to pick up all the sticks; otherwise, we follow the order of each stick produced by Topological Sort algorithm to pick them up.

Pseudocode:

```
// returns the order of sticks picked up or if it wasn't possible
PickUpSticks(s(1), s(2), ..., s(n)) //s(n)=listofnsticks

    for(each stick s in list s(v))
        Construct a vertex v for Graph G;

    Project all sticks to x-y plane;
    Find all intersections between any two sticks by running the
    algorithm from(a) between every pair of sticks;

    if(spatial relation between two sticks)
        add directed edge between vertices of two sticks to G;
```

```

G.topsort(); // topsort() method from Addison Wesley textbook

if cycle is found in G by topsort()
    return No;

else
    return the order of each stick produced by TopologicalSort;

```

Problem 3 (25pts)

Solution

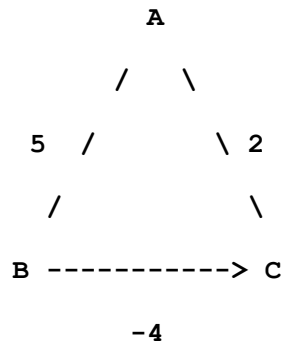
Dijkstra's Table:

Time	visit	A		B		C		D		E		F		PQ
		cost	p	cost	p	cost	p	cost	p	cost	p	cost	p	
0		0		inf		inf		inf		inf		inf		A
1	A	0		2	A	9	A	inf		inf		inf		BC
2	B	0		2	A	8	B	5	B	10	B	inf		D C E
3	D	0		2	A	8	B	5	B	7	D	6	D	F E C
4	F	0		2	A	7	F	5	B	7	D	6	C	E C
5	E (or C)	0		2	A	7	F	5	B	7	D	6	C	C
6	C (or E)	0		2	A	7	F	5	B	7	D	6	C	

The path is of length 7 from A to E with path of A -> B -> D -> E

Problem 4 (25pts)

Solution



$V = \{A, B, C\}$ $E = \{(A, B), (A, C), (B, C)\}$

Start Dijkstra's at A.

Dijkstra's Algorithm will update the cost for B to 5 and C to 2. Then it will visit C, locking in path A \rightarrow C with cost 1. However, A \rightarrow B \rightarrow C is shorter (cost $5 - 4 = 1$), but will be missed by the algorithm.