Ph.D. Qualifying Exam: Data Structures and Algorithms

This is a closed book exam. The total score is 105 points. Please answer all questions.

1. Given two sorted arrays A and B in non-descending order, your goal is to find the median of their union. We use $|A|$ and $|B|$ to represent the lengths of each array, respectively. When the union contains an even number of numbers, please find the lower median. (DPV Chapter 2)

(20 points) (a) Design an iterative algorithm that runs in $\Theta(|A|+|B|)$. No recursive calls are allowed. You must provide complete pseudocode with input, output, and steps that may include loops.

Solution:

```
function median-iterative(A, B)
C = merge(A, B)
p = ⌊(|A| + |B| + 1)/2⌋
return C[p]
```

```
function merge(x[1...k], y[1...l])
if k == 0: return y[1...l]
if l == 0: return x[1...k]
allocate an array z of length k + l
m = 1, i = 1, j = 1
while i ≤ k and j ≤ l
    if x[i] < y[j]:
      z[m] = x[i]
      i = i + 1
    else:
      z[m] = y[j]
      j = j + 1
    m = m + 1
if i ≤ k: z[m...(l + k)] = x[i...k]
if j ≤ l: z[m...(l + k)] = x[j...l]
return z
```

(15 points) (b) Design a recursive algorithm that runs in $O(\lg(|A| \cdot |B|))$ employing divide-and-conquer.

Solution:

```
function median-recursive(A, B)
p = ⌊(|A| + |B| + 1)/2⌋
return select(p, A, 1, |A|, B, 1, |B|)
```

---

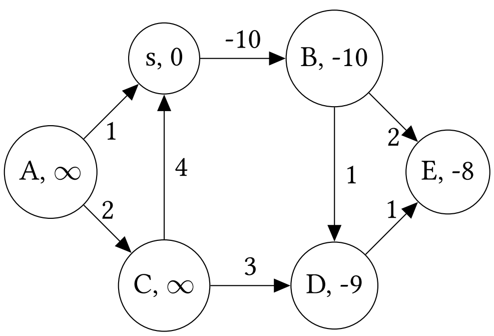
function select(p, A, i_1, j_1, B, i_2, j_2)
if $p == 1$, return $\min\{A[i_1], B[i_2]\}$
$s_1 = i_1 + \lfloor p/2 \rfloor - 1$
$s_2 = i_2 + \lfloor p/2 \rfloor - 1$
if $A[s_1] < B[s_2]$
  $u =$ select($p - \lfloor p/2 \rfloor$, A, $s_1 + 1$, j_1, B, i_2, s_2)
else
  $u =$ select($p - \lfloor p/2 \rfloor$, A, i_1, s_1, B, $s_2 + 1$, j_2)
return u

---

2. Let G be a directed graph where edges leaving the source node s all have negative weights. All other edges in the graph are positively weighted. (DPV Chapter 4)

(20 points) (a) Draw such a graph that Dijkstra's algorithm correctly finds all shortest paths from the source node s. Please label distances from s to each node in the graph.

Solution:

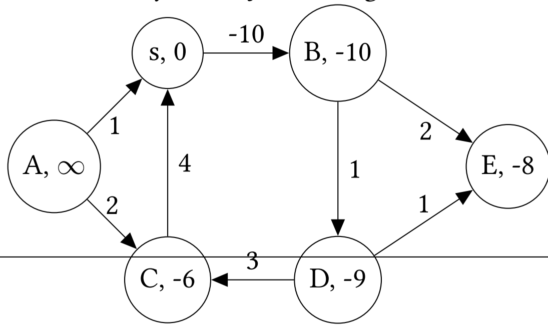


(15 points) (b) Does Dijkstra's algorithm always work generally? If yes, prove your claim; otherwise, give a counter example and point errors by Dijkstra's algorithm.

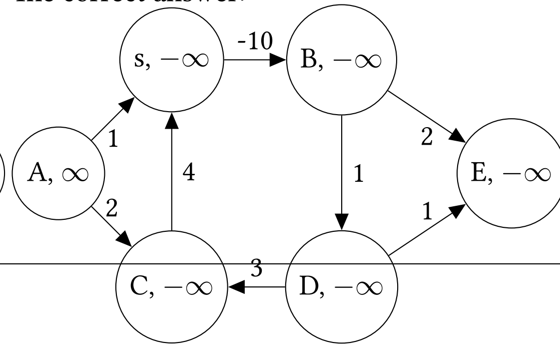
Solution:

If negative cycles are found in a graph, then Dijkstra's algorithm does not work. Here is a counter example.

A solution by the Dijkstra's algorithm:



The correct answer:



If the graph does not contain a negative cycle, Dijkstra's algorithm works correctly to identify shortest paths from source node s.

Proof. Let w be the most negative weight of an edge outgoing from the source node s in graph G. Add w to all negative edges from s to obtain a new graph G'. A shortest path π from s to u in G' must be a shortest path from s to u in G. This is because lengths of all paths from s are longer by the same amount of w in G' than G, if there is no negative cycle in the graph. If we run Dijkstra's algorithm on G, all nodes will be processed exactly in the same way as on G' because relative positions of nodes in the priority queue are identical during the run. Thus the algorithm works on such a graph as long as it does not have a negative cycle. □

3. Let X and Y be two strings. Please give the recurrence equations on finding the edit distance between X and Y by dynamic programming. We define the edit distance as the minimal cost of operations including substitution, insertion, and deletion to transform X to Y. (DPV Chapter 6)

(20 points) (a) If the cost of substitution, insertion, or deletion is all 1, give a recurrence equation for the edit distance between prefixes $X[1...i]$ and $Y[1...j]$.

Solution:

$$d[i,j] = \min \begin{cases} d[i-1,j-1] & X[i] = Y[j] \\ d[i-1,j-1]+1 & X[i] \neq Y[j] \\ d[i-1,j]+1 \\ d[i,j-1]+1 \end{cases} \tag{1}$$

(15 points) (b) To allow a general distance, we define the cost of substitution $X[i]$ by $Y[j]$ by $\delta(X[i], Y[j])$, the cost of insertion of $Y[j]$ to X by $\delta(-, Y[j])$, and the cost of deletion of $X[i]$ from X by $\delta(X[i], -)$, respectively. Please give a recurrence equation for the edit distance between prefixes $X[1...i]$ and $Y[1...j]$.

Solution:

$$d[i,j] = \min \begin{cases} d[i-1,j-1] + \delta(X[i], Y[j]) \\ d[i-1,j] + \delta(X[i], -) \\ d[i,j-1] + \delta(-, Y[j]) \end{cases} \tag{2}$$