Exam Statistics

111 students took the exam. The range of scores was 0–95, with a mean of 45.85, a median of 46, and a standard deviation of 20.18.

Problem Solutions

1. Fibonacci Heaps

   (a) Tall-Heap(1, b) clearly produces

   \[
   \begin{array}{c}
   b \\
   \end{array}
   \]

   because line 1 creates an empty Fibonacci heap, after which line 2 inserts \( b \). Tall-Heap(2, b) creates an empty Fibonacci heap \( T \) in line 5, after which lines 6–8 insert \( b - 1 \), \( b \), and \( b + 1 \), respectively, all singleton nodes at the root level. The \textsc{Extract-Min}(T) deletes \( b - 1 \), but then the consolidation phase then combines the two remaining roots each have degree 0 into

   \[
   \begin{array}{c}
   b \\
   b + 1 \\
   \end{array}
   \]

   That takes care of the base cases. Now assume that Tall-Heap\((h, b)\) produces
Tall-Heap\((h+1, b)\), for \(h + 1 > 2\) first calls Tall-Heap\((h, b + 1)\) producing the heap

by induction. Inserting \(b - 0.5\), \(b\), and \(b + 0.5\) adds these three values as roots of degree 0. Extracting the minimum deletes \(b - 0.5\), and the consolidation phase first merges degree 0 roots \(b\), and \(b + 0.5\) into

and then combines the two roots of degree 1 to form
whereupon deleting \( b + 0.5 \) produces

\[
\begin{array}{c}
\bullet \\
\downarrow \\
\bullet \\
\downarrow \\
\vdots \\
\downarrow \\
\bullet
\end{array}
\]

as desired.

(b) Although generally both extracting the minimum or deleting an element require amortized time \( O(\log n) \) in a Fibonacci heap of \( n \) elements, here the consolidation phase takes only \( O(1) \) worst-case time; making the heap initially and doing the insertions are all \( O(1) \) worst-case time. Thus \( \text{Tall-Heap}(h + 1, b) \) takes time \( O(h) \).

(c) By part (a), \( \text{Tall-Heap}(n, 1) \) produces

\[
\begin{array}{c}
1 \\
\downarrow \\
2 \\
\downarrow \\
3 \\
\downarrow \\
\vdots \\
\downarrow \\
n
\end{array}
\]
2. Depth First Search

function DFS-visit(u)
1: color[u] ← GRAY
2: d[u] ← time ← time + 1
3: for all v ∈ Adj[u] do
4: if color[v] = WHITE then
5: Tree edge
6: π[v] ← u
7: DFS-visit(v)
8: else if color[v] = GRAY then
9: Back edge
10: else if d[u] < d[v] then
11: Forward edge
12: else
13: Cross edge
14: end if
15: end for
16: color[u] ← BLACK
17: f[u] ← time ← time + 1

3. Shortest Paths

(a) Consider two paths: \( P_1 = (u, u_1, u_2, \ldots, v) \) and \( P_2 = (u, v_1, v_2, \ldots, v) \) of equal length, \( L = |P_1| = |P_2| \). Under the transformation, the path length telescopes:

\[
|P_1'| = \sum_{\text{edge } (a,b) \in P_1} \frac{w_V(a) + w_V(u_b)}{2} = |P_1| - \frac{w_V(u) + w_V(v)}{2} = L - \frac{w_V(u) + w_V(v)}{2}
\]

and

\[
|P_2'| = \sum_{\text{edge } (a,b) \in P_2} \frac{w_V(a) + w_V(u_b)}{2} = |P_2| - \frac{w_V(u) + w_V(v)}{2} = L - \frac{w_V(u) + w_V(v)}{2}
\]

Thus, equal paths in the vertex-weighted graph also have equal path lengths in the edge-weighted graph.

(b) Reasoning as in part (a), the newly weighted paths differ by:

\[
|P_2'| - |P_1'| = |P_2| - \frac{w_V(u) + w_V(v)}{2} - |P_1| = |P_2| - |P_1|
\]

So that if \( P_1 \) is shorter than \( P_2 \) in the vertex-weighted graph, it will also be shorter in the edge-weighted graph.
4. NP-Completeness

The EXAM3-SCHEDULING decision problem is “Given an \( n \times n \) graph job compatibility matrix \( C_{ij} \) and a constant \( d \), can the \( n \) jobs be scheduled in \( d \) days?”

EXAM3-SCHEDULING is clearly in the class NP because given \( C_{ij} \) and a proposed schedule using \( d \) days, we can easily check in time \( O(n^2) \) whether the schedule has two incompatible jobs scheduled for the same day.

To prove EXAM3-SCHEDULING is NP-hard, we reduce from GRAPH-COLORING (Problem 34-3 on page 1103 in CLRS; discussed at length in the lectures of April 11–13 and in HW 8). Given a graph \( G = (V, E) \) and an integer \( k > 0 \), we can determine whether \( G \) can be \( k \)-colored by using the \( |V| \times |V| \) Boolean matrix which is the adjacency matrix for \( G \) (see page 591 of CLRS or the lecture notes from March 28) as a compatibility matrix for \(|V| \) jobs and \( k \) as the number of days. If the jobs can be scheduled, assign a unique color to each of the \( k \) days and color each job (vertex) with a color of the day on which it is scheduled. Similarly, if the graph can be colored with \( k \) colors, the coloring gives a \( k \)-day schedule for the \(|V| \) jobs.

5. Spanning Tree Approximation

(a) The following graph achieves the desired result for any \( k > 0 \):

```

A
  /\  2k
 /   \
B----C
  \
   

```

The minimum spanning tree MST has edges \( AB \) and \( BC \) with cost 2. But the stupid algorithm could construct the spanning tree ST consisting of \( AC \) and \( BC \) with cost \( 2k + 1 \). Now

\[
\frac{|ST|}{|MST|} = \frac{2k + 1}{2} > k.
\]

(b) The new approximation bound is clearly 2 because any spanning tree has \(|V| - 1\) edges, and hence the MST has cost at least \(|V| - 1\). But no edge costs more than 2, so the stupid tree will have cost at most \( 2|V| - 2 \). Thus

\[
\frac{|ST|}{|MST|} \leq \frac{2}{1} = 2.
\]