CSE 421 Section 6

Midterm Review

Announcements & Reminders

- **HW4** regrade requests are open, answer keys on Ed
- **HW5** was due yesterday, 10/30
  - Late submissions open until tomorrow, 11/1 @ 11:59pm
- There is no homework this week.
- Your **midterm exam** is on **Monday, 11/4 @ 6:00–7:30pm, Gates G04**
  - Let us know by **tomorrow, Friday 11/1** if you cannot make it
  - If you are sick, let us know as soon as you know
  - A **practice midterm** is available on Ed

Midterm format

- Several multiple choice/short answer problems
- 3 long-form problems
  - Similar in style to homework
- 90 minutes
- You will be given a standard reference sheet, view it on Ed
- You may bring one sheet of double sided 8.5x11” paper containing your own handwritten notes.
  - Must write name, student number, and UW NetID
  - Must turn in with exam

Today's plan

1. (35 min) 6 stations around the room with practice problems
    - Station 1: Short answer
    - Station 2: Stable matching reduction*
    - Station 3: Graph algorithms
    - Station 4: Greedy algorithms*
    - Station 5: Divide and conquer*
    - Station 6: Dynamic programming
2. (10 min) Go over some of these problems

**the problem at this station was an extra problem on a previous section handout*

Problems

1	2	3	4	5	6

Problem 1 – Short answer

If p ranks r first and r ranks p first, then (p, r) must be in every stable matching.

Solution

Problem 1 – Short answer

If p ranks r first and r ranks p first, then (p, r) must be in every stable matching.

True. If p and r were not matched, then they prefer each other over the current matches, so this is an instability.

Problem 1 – Short answer

Running DFS on a directed acyclic graph may produce:

- ❑ Tree edges
- ❑ Back edges
- ❑ Forward edges
- ❑ Cross edges

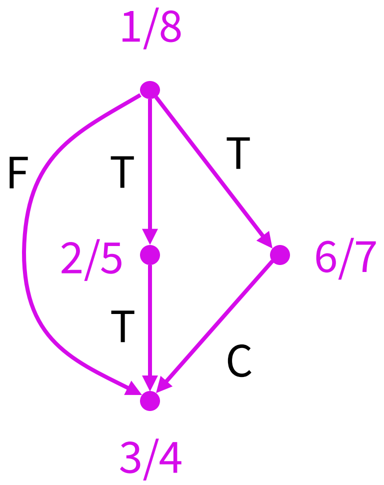
Solution

Problem 1 – Short answer

Running DFS on a directed acyclic graph may produce:

- ❑ Tree edges
- ❑ Back edges
- ❑ Forward edges
- ❑ Cross edges

All except back edges, since they create cycles.



Problem 1 – Short answer

The recurrence $T(n) = 2T(n/3) + \Theta(n^2)$ simplifies to…?

Solution

Problem 1 – Short answer

The recurrence $T(n) = 2T(n/3) + \Theta(n^2)$ simplifies to…?

$\Theta(n^2)$. By master theorem, since $2 < 3^2$.

Problem 1 – Short answer

Suppose G has positive, distinct edge costs. If T is an MST of G, then it is still an MST after replacing each edge cost c_e with c_e^2.

Solution

Problem 1 – Short answer

Suppose G has positive, distinct edge costs. If T is an MST of G, then it is still an MST after replacing each edge cost c_e with c_e^2.

True. Kruskal's (or Prim's) only depends on the relative order of edge costs. Furthermore, because costs are distinct, there is a unique MST, so Kruskal's algorithm found T before and will still find T now.

Problem 1 – Short answer

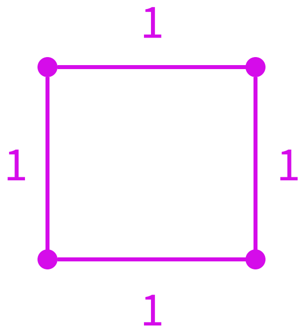
Let $G = (V, E)$ be a weighted, undirected graph. Consider any cut $S \subseteq V$, and let e be an edge of minimum weight across the cut S. Then every MST contains e.

Solution

Problem 1 – Short answer

Let $G = (V, E)$ be a weighted, undirected graph. Consider any cut $S \subseteq V$, and let e be an edge of minimum weight across the cut S. Then every MST contains e.

False. The theorem requires edge weights be distinct. Consider:



Return to problem select

Problem 2 – Stable matching reduction

There are R riders, H horses with $2H < R < 3H$. Riders and horses have preferences for each other. Also, riders prefer the first 2 rounds. Horses prefer to ride every round.

Set up 3 rounds of rides, so that every rider will ride a horse exactly once, every horse does exactly 2 or 3 rides, and there are no unstable matches.

Solution

Problem 2 – Stable matching reduction

There are R riders, H horses with $2H < R < 3H$. Riders and horses have preferences for each other. Also, riders prefer the first 2 rounds. Horses prefer to ride every round.

Set up 3 rounds of rides, so that every rider will ride a horse exactly once, every horse does exactly 2 or 3 rides, and there are no unstable matches.

For all horses h, create h_1, h_2, and h_3. Add $3H - R$ dummy riders. For preference lists:

- For real riders: original list with h_1 and h_2 replacing h, then original list with h_3's.
- For dummy riders: all h_3 (in any order), then everything else (in any order).
- For horse-in-rounds: original list, then dummy riders in any order.

Solution

Problem 2 – Stable matching reduction

For all horses h, create h_1, h_2, and h_3. Add $3H - R$ dummy riders. For preference lists:

- For real riders: original list with h_1 and h_2 replacing h, then original list with h_3's.
- For dummy riders: all h_3 (in any order), then everything else (in any order).
- For horse-in-rounds: original list, then dummy riders in any order.

Then:

- Every rider is matched because library returns perfect matching.
- Dummy matched to horse in round 1 or 2 is unstable.
- Horse and real rider who prefer each other is unstable.

Return to problem select

Problem 3 – Graph modeling

Given $(a_1, b_1), \ldots, (a_n, b_n)$, the person living in unit a_i is moving to b_i. Some people may be new arrivals ($a_i = \text{null}$) or moving out ($b_i = \text{null}$). Give an algorithm that returns a valid moving order (every unit is vacated before someone moves in), or "not possible" and a minimal list of pairs that explains why.

Solution

Problem 3 – Graph modeling

Given $(a_1, b_1), \ldots, (a_n, b_n)$, the person living in unit a_i is moving to b_i. Some people may be new arrivals ($a_i = \text{null}$) or moving out ($b_i = \text{null}$). Give an algorithm that returns a valid moving order (every unit is vacated before someone moves in), or "not possible" and a minimal list of pairs that explains why.

$$(2, \text{null}) \rightarrow (1, 2) \rightarrow (\text{null}, 1)$$

$A \rightarrow B$ iff A must happen before B

$$\begin{tikzcd} (3,4) \arrow[r, bend left] & (4,3) \arrow[l, bend left] \end{tikzcd}$$

Solution

Problem 3 – Graph modeling

Given $(a_1, b_1), \ldots, (a_n, b_n)$, the person living in unit a_i is moving to b_i. Some people may be new arrivals ($a_i = \text{null}$) or moving out ($b_i = \text{null}$). Give an algorithm that returns a valid moving order (every unit is vacated before someone moves in), or "not possible" and a minimal list of pairs that explains why.

$$(2, \text{null}) \rightarrow (1, 2) \rightarrow (\text{null}, 1)$$

$$\begin{tikzcd}(3,4) \arrow[r, bend left] & (4,3) \arrow[l, bend left]\end{tikzcd}$$

1. Check for cycles with B/DFS.
    a. If there is a cycle, not possible.

Solution

Problem 3 – Graph modeling

Given $(a_1, b_1), \dots, (a_n, b_n)$, the person living in unit a_i is moving to b_i. Some people may be new arrivals ($a_i = \text{null}$) or moving out ($b_i = \text{null}$). Give an algorithm that returns a valid moving order (every unit is vacated before someone moves in), or “not possible” and a minimal list of pairs that explains why.

$$(2, \text{null}) \rightarrow (1, 2) \rightarrow (\text{null}, 1)$$

$$(3, 5) \longrightarrow (4, 3)$$

1. Check for cycles with B/DFS.
   a. If there is a cycle, not possible.
   b. If there is no cycle, topo sort.

Return to problem select

Problem 4 – Greedy algorithms

Given a set $\mathcal{X}$ of integer intervals $[a, b] \subseteq \mathbb{Z}$, find the smallest set $\mathcal{Y} \subseteq \mathcal{X}$ such that every point in any interval of $\mathcal{X}$ belongs to some interval of $\mathcal{Y}$ (i.e. $\mathcal{Y}$ covers $\mathcal{X}$).

Solution

Problem 4 – Greedy algorithms

Given a set $\mathcal{X}$ of integer intervals $[a, b] \subseteq \mathbb{Z}$, find the smallest set $\mathcal{Y} \subseteq \mathcal{X}$ such that every point in any interval of $\mathcal{X}$ belongs to some interval of $\mathcal{Y}$ (i.e. $\mathcal{Y}$ covers $\mathcal{X}$).

Repeatedly pick the interval with the largest end point that covers the smallest yet-uncovered point.

(For implementation details, see solutions tonight. Naively finding the "smallest yet-uncovered point" is technically correct but slow.)

Solution

Problem 4 – Greedy algorithms

Repeatedly pick the interval with the largest end point that covers the smallest yet-uncovered point.

Proof sketch: (greedy stays ahead)

- We output $[a_1, b_1], \ldots, [a_k, b_k]$ and suppose $[o_1, p_1], \ldots, [o_l, p_l]$ is valid and sorted.
- Can prove by induction that $b_i \geq p_i$ for all i (explain why this is enough).
  - After selecting $[a_1, b_1], \ldots, [a_{i-1}, b_{i-1}]$ the smallest uncovered point is larger than b_{i-1} and hence not covered by $[o_1, p_1], \ldots, [o_{i-1}, p_{i-1}]$ by induction.
  - If $[o_i, p_i]$ does not cover it, by sortedness, other solution is invalid.
  - If $[o_i, p_i]$ does cover it, then $b_i \geq p_i$ because that was our greedy criterion.

Return to problem select

Problem 5 – Divide and conquer

$A[1..n]$ is a mountain if there is a peak i such that

$$A[1] < \cdots < A[i-1] < A[i] \text{ and } A[i] > A[i+1] > \cdots > A[n].$$

The peak may be at 1 or n. Given a mountain, find the peak in $O(\log n)$ time.

Solution

Problem 5 – Divide and conquer

$A[1..n]$ is a mountain if there is a peak i such that

$$A[1] < \cdots < A[i-1] < A[i] \text{ and } A[i] > A[i+1] > \cdots > A[n].$$

The peak may be at 1 or n. Given a mountain, find the peak in $O(\log n)$ time.

function peakFinder(i, j) (base case omitted for slide brevity)

1. $m \leftarrow \left\lfloor \frac{i+j}{2} \right\rfloor$
2. **if** $A[m+1]$ exists and $m+1 \leq j$ and $A[m] < A[m+1]$ (checking for edge cases)
   a. **return** peakFinder($m+1, j$)
3. **else if** $A[m-1]$ exists and $i \leq m-1$ and $A[m-1] > A[m]$
   a. **return** peakFinder($i, m-1$)
4. **else return** m

Solution

Problem 5 – Divide and conquer

function peakFinder(i, j)

1. $m \leftarrow \left\lfloor \frac{i+j}{2} \right\rfloor$
2. **if** $A[m+1]$ exists and $m+1 \leq j$ and $A[m] < A[m+1]$
   a. **return** peakFinder$(m+1, j)$
3. **else if** $A[m-1]$ exists and $i \leq m-1$ and $A[m-1] > A[m]$
   a. **return** peakFinder$(i, m-1)$
4. **else return** m

Induction on k:

For all i and j with $j - i = k$, **if $A[i..j]$ contains the peak**, peakFinder(i, j) finds it.

(crucial point!)

Solution

Problem 5 – Divide and conquer

Induction on k:

For all i and j with $j - i = k$, **if $A[i..j]$ contains the peak**, peakFinder(i, j) finds it.

Three cases for where the peak is:

1. The peak is in $A[m+1..j]$.
   - We end up in the first if branch (explain why).
   - Can apply IH to peakFinder$(m+1, j)$ because the peak is in $A[m+1..j]$!
2. The peak is in $A[i..m-1]$. Similar.
3. The peak is $A[m]$.
   - We end up in the else branch (explain why).

Return to problem select

Problem 6 – Dynamic programming

Compute the maximum reward going from $(1, 1)$ to (m, n) on a grid, where you gain $R[i, j]$ whenever passing through (i, j). Starting/ending count as passing through. $R[i, j]$ may be negative (penalty) or $-\infty$ (impassible).

Solution

Problem 6 – Dynamic programming

Compute the maximum reward going from $(1,1)$ to (m,n) on a grid, where you gain $R[i,j]$ whenever passing through (i,j). Starting/ending count as passing through. $R[i,j]$ may be negative (penalty) or $-\infty$ (impassible).

$$\mathrm{OPT}(i,j) = R[i,j] + \max\big(\mathrm{OPT}(i-1,j), \mathrm{OPT}(i,j-1)\big) \qquad i,j > 2$$

$$\mathrm{OPT}(1,1) = R[1,1]$$
$$\mathrm{OPT}(1,j) = R[1,j] + \mathrm{OPT}(1,j-1) \qquad j > 2$$
$$\mathrm{OPT}(i,1) = R[i,1] + \mathrm{OPT}(i-1,1) \qquad i > 2$$

Return to problem select