

Lecture10



CSE 417

Algorithms and Complexity

Autumn 2024

Lecture 10 – Greedy Algorithms III

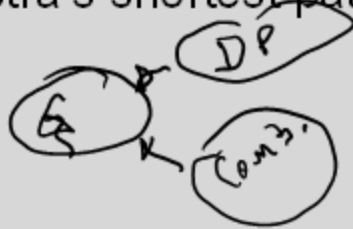
Announcements

- **Today's lecture**
 - Kleinberg-Tardos, 4.3, 4.4
- **Friday**
 - Kleinberg-Tardos, 4.4, 4.5
- **Text book has lots of details on some of the proofs that I cover quickly**



Greedy Algorithms

- Solve problems with the simplest possible algorithm
- Today's problems (Sections 4.3, 4.4)
 - Another homework scheduling task
 - Optimal Caching
- Start Dijkstra's shortest paths algorithm



$$\begin{array}{r} \underline{2} \quad \underline{1} \\ 10 \quad \underline{1} \\ \hline \quad \quad \underline{3} \end{array}$$

3

Scheduling Theory

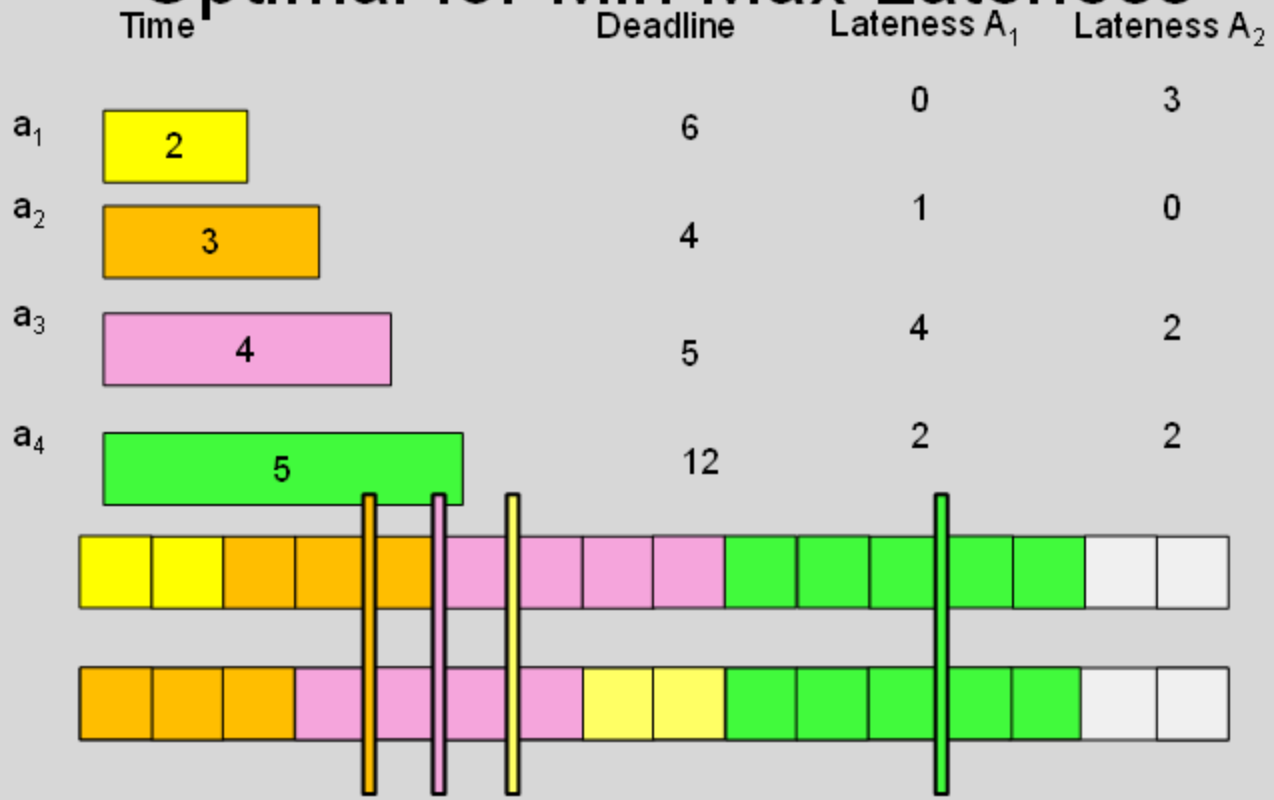
- **Tasks**
 - Execution time, value, release time, deadline
- **Processors**
 - Single processor, multiple processors
- **Objective Function – many options, e.g.**
 - Maximize tasks completed
 - Minimize number of processors to complete all tasks
 - Minimize the maximum lateness
 - Maximize value of tasks completed by deadline

Homework Scheduling

- Each task has a length t_i and a deadline d_i
- All tasks are available at the start
- One task may be worked on at a time
- All tasks must be completed

- Goal minimize maximum lateness
 - Lateness: $L_i = f_i - d_i$ if $f_i \geq d_i$

Result: Earliest Deadline First is Optimal for Min Max Lateness



Another version of HW scheduling

- Assign values to HW units
- Maximize value completed by deadlines
- Simplifying assumptions
 - All Homework items take one unit of time
 - All items available at time 0
 - Each item has an integer deadline
 - Each item has a value
 - Maximize value of items completed before their deadlines

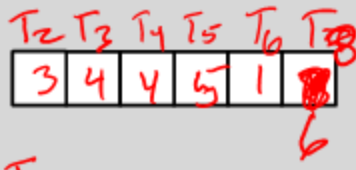
| T | D | V |
|--------------------------|--------------|--------------|
| T₁ | 1 | 1 |
| T ₂ | 2 | 2 |
| T ₃ | 2 | 2 |

$D=6$

Example

| Task | Value | Deadline |
|----------------|-------|----------|
| T ₁ | 2 | 2 |
| T ₂ | 3 | 2 |
| T ₃ | 4 | 4 |
| T ₄ | 4 | 4 |
| T ₅ | 5 | 4 |
| T ₆ | 1 | 6 |
| T ₇ | 1 | 6 |
| T ₈ | 6 | 6 |

RT
4
4
2
2
2
0
0
0



What is the maximum value of tasks you can complete by their deadlines?

What do you do first?

8



Problem transformation

- Convert to an equivalent problem with release times and a uniform deadline
- If D is the latest deadline, set r'_i as $D - d_i$ and d'_i as D

Greedy Algorithm

- Starting from $t = 0$, schedule the highest value available task

```
S =  $\emptyset$ ;  
for i = 0 to D - 1  
    Add tasks with release time i to S;  
    Remove highest value task t from S;  
    Schedule task t at i;
```

Correctness argument

- Show that the item at $t = 1$ is scheduled correctly
 - The argument can be repeated for $t=2, 3, \dots$
 - Or the argument can be put in the framework of mathematical induction

First item scheduled is correct

- Let t be the task scheduled at $i = 1$, then there exists an optimal schedule with t at $i = 1$
 - Suppose $\text{Opt} = \{a_1, a_2, a_3, \dots\}$ is an optimal schedule:
 - Case 1: $t = a_1$
 - Case 2: $t \notin \text{Opt}$
 - Case 3: $t \neq a_1$ and $t \in \text{Opt}$
- Handwritten notes:*
 - A red circle around a_1 and a red box around a_3 in the set notation.
 - A red arrow points from the box around a_3 to the letter t above it.
 - Red text: "replace a_1 by t " with a checkmark next to Case 2.
 - Red text: "Reorder" with a checkmark next to Case 3.

Interpretation

- The transformation was done so that we could think about the first item to schedule, as opposed to the last item to schedule
- In the original problem with deadlines, this is asking “what task do I do last”
 - So this is a procrastination based approach!

Optimal Caching



- **Memory Hierarchy**
 - Fast Memory (RAM)
 - Slow Memory (DISK)
 - Move big blocks of data from DISK to RAM for processing
- **Caching problem:**
 - Maintain collection of items in local memory
 - Minimize number of items fetched

Farthest in future

Caching example

| | | |
|--------------|---|--------------|
| A | B | C |
|--------------|---|--------------|

~~C~~
A

~~D~~
E
D

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

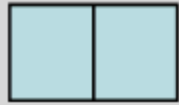
A, B, C, D, ~~A~~, E, ~~B~~, ~~A~~, D, ~~A~~, C, ~~B~~, ~~D~~, A

Optimal Caching

- If you know the sequence of requests, what is the optimal replacement pattern?
- Note – it is rare to know what the requests are in advance – but we still might want to do this:
 - Some specific applications, the sequence is known
 - Register allocation in code generation
 - Competitive analysis, compare performance on an online algorithm with an optimal offline algorithm

Farthest in the future algorithm

- Discard element used farthest in the future



A, B, C, A, C, D, C, B, C, A, D

Correctness Proof

- Sketch
- Start with Optimal Solution O
- Convert to Farthest in the Future Solution $F-F$
- Look at the first place where they differ
- Convert O to evict $F-F$ element
 - There are some technicalities here to ensure the caches have the same configuration . . .

Dijkstra



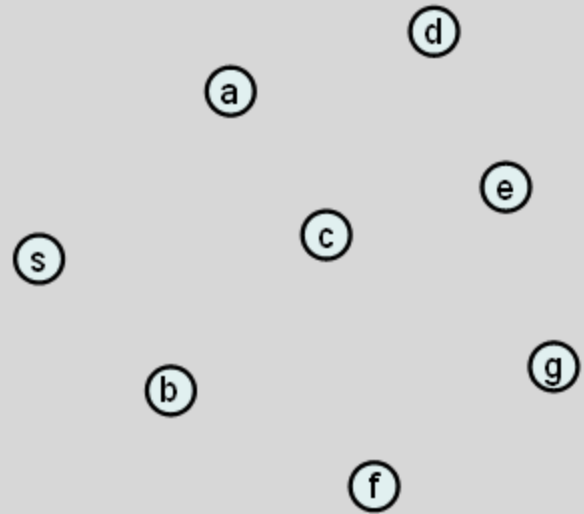
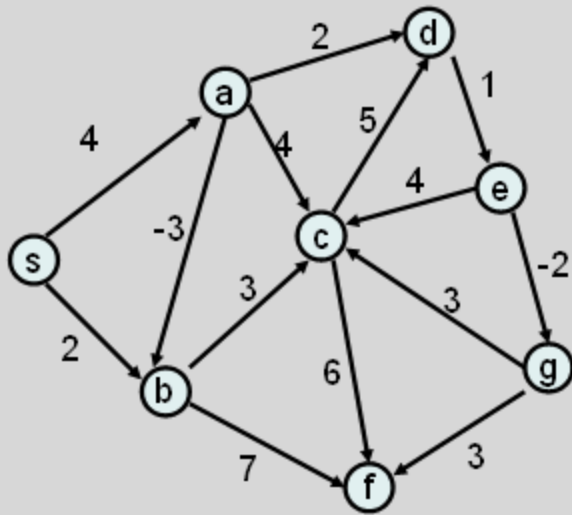
Single Source Shortest Path Problem

- Given a graph and a start vertex s
 - Determine distance of every vertex from s
 - Identify shortest paths to each vertex
 - Express concisely as a “shortest paths tree”
 - Each vertex has a pointer to a predecessor on shortest path



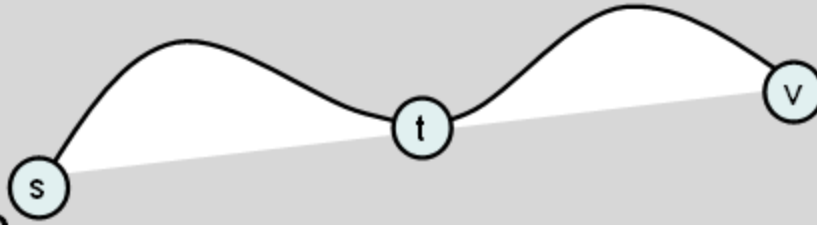
20

Construct Shortest Path Tree from s



Warmup

- If P is a shortest path from s to v , and if t is on the path P , the segment from s to t is a shortest path between s and t



- WHY?

Assume all edges have non-negative cost

Dijkstra's Algorithm

$S = \{ \}$; $d[s] = 0$; $d[v] = \text{infinity}$ for $v \neq s$

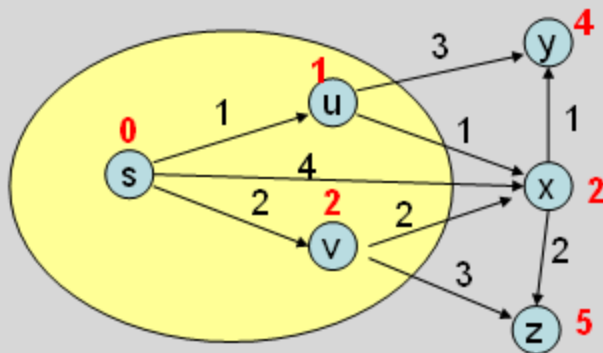
While $S \neq V$

 Choose v in $V-S$ with minimum $d[v]$

 Add v to S

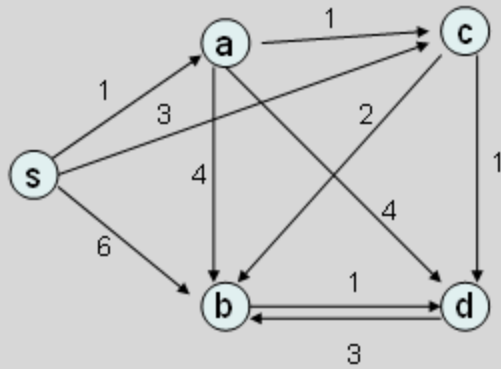
 For each w in the neighborhood of v

$d[w] = \min(d[w], d[v] + c(v, w))$



23

Simulate Dijkstra's algorithm (starting from s) on the graph



| Round | Vertex Added | s | a | b | c | d |
|-------|--------------|---|---|---|---|---|
| | | | | | | |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |