P THINK FAST S

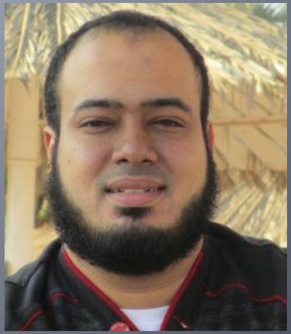
Competitive Programming

From Problem 2 Solution in O(1)

Greedy Algorithms

Stay Ahead Technique

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University



Proving Greedy

- We may prove greedy properties directly
  - Last session
- 2 popular techniques can be used for proving
  - Stay Ahead [Today Session]
  - Exchange Argument [Next Session]
- And some other ways

Proof technique: Greedy Stays Ahead

- Can we prove that step-by-step, our greedy algorithm is "**at least as good**" than any other optimal solution (e.g. always <= or >=)
- If so, we can show greedy optimality!
- To administer such comparison, we need some **measure** to compare
- More suitable when optimal solutions **vary in length** (e.g. shortest paths of different length)

Main 4 steps

- Label the partial solutions
  - Both your greedy **A** and a given optimal one **O**
- Define a measure
  - The hardest part. It can be one or **more** measure
  - E.g. <u>Among first j items</u>: Get finish time, total weight, largest index, ..etc
- Prove greedy stay ahead: By induction
  - For all r <= k: F(a1, a2...ar) <= or >= F(o1, o2...or)
- Prove optimality: By contradiction
  - Assume greedy not optimal...but it stays ahead!

Recall: Interval scheduling

Interval scheduling

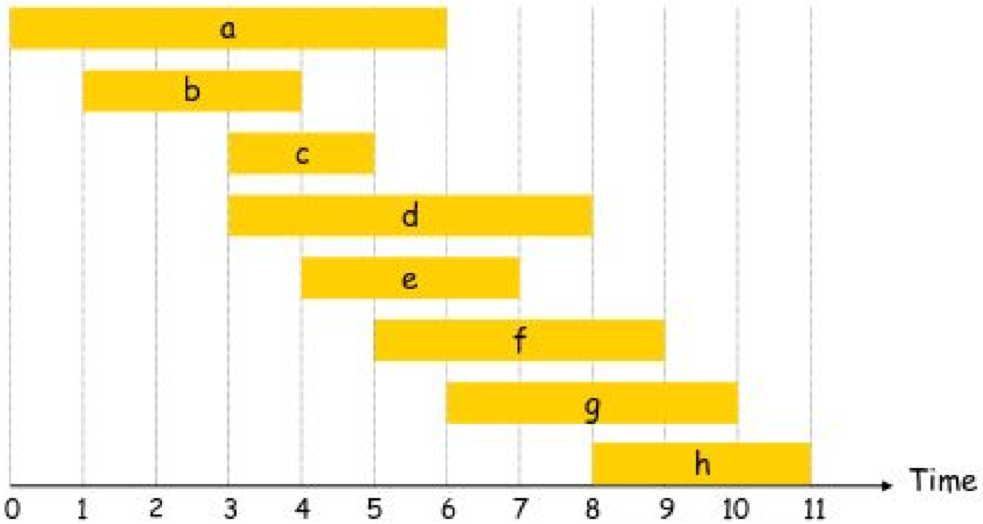
- Formally
  - Requests $1,2,\ldots,n$
    - request i has start time s_i and finish time $f_i > s_i$
  - Requests i and j are **compatible** iff either
    - request i is for a time entirely before request j
      - $f_i \leq s_j$
    - or, request j is for a time entirely before request i
      - $f_j \leq s_i$
  - Set A of requests is **compatible** iff every pair of requests $i,j \in A$, $i \neq j$ is compatible
  - Goal: Find maximum size subset A of compatible requests

Src: https://courses.cs.washington.edu/courses/cse421/08au/Greedy.pdf

Recall: Interval scheduling

Interval Scheduling

- Interval scheduling.
  - Job j starts at s_j and finishes at f_j.
  - Two jobs compatible if they don't overlap.
  - Goal: find maximum subset of mutually compatible jobs.



Src: https://courses.cs.washington.edu/courses/cse421/08au/Greedy.pdf

Recall: Interval scheduling

Greedy Algorithm for Interval Scheduling

R$\leftarrow$set of all requests

A$\leftarrow\varnothing$

While R$\neq\varnothing$ do

  Choose request $\mathbf{i}\in\mathbf{R}$ with smallest finishing time $\mathbf{f_i}$

  Add request **i** to **A**

  Delete all requests in **R** that are not compatible with request **i**

Return **A**

Src: https://courses.cs.washington.edu/courses/cse421/08au/Greedy.pdf

Interval scheduling

- Any optimal solution will have **same length**
- Our greedy based on **finish times**, so one way to think about measure is finish time
- E.g. we need to prove **greedy finish time is superior** and provide more space for elements
- Concern: Can't compare with given optimal solution **o** as it is not sorted!
  - Cool, finish optimal solution by finish time

Interval scheduling: Labeling

- Let A = {i1, i2….ik}: Greedy output
- Let O = {j1, j2….jk}: Some optimal solution
  ordered by finish time

Interval scheduling: Measure

- Let S be some selected jobs so far
- LF(S) = Last finish time of S
  - LF(i1, i2….ir) = F[ir] … e.g. finish of last tiem
  - LF(j1, j2….jr) = F[jr] … e.g. finish of last tiem
- Goal, for all r <= k
  - Show: LF(i1, i2….ir) <= LF(j1, j2….jr)
  - Or simply
  - Show: F[ir] <= F[jr]

Interval scheduling: Stay Ahead

- By induction
- For r = 1, F[i1] <= F[j1]
  - By algorithm nature, our first task, smallest in finish time
- For r > 1
  - Assume true for r-1, then F[i_r-1] <= F[j_r-1]
  - Then if some job K can be added to the given optimal solution, it can be added to greedy too, as greedy is a smaller in overall finish time.

Interval scheduling: Optimality

- We proved for all r <= k: F[ir] <= F[jr]
  - Then F[ik] <= F[jk]   : last item finish time
- Assume A is not optimal, then O has extra item(s) not in A
  - As A stays ahead and O is sorted, this item starts after finish time of last job, F[jk]
  - However, as A stays ahead, this item is compatible with greedy too and can be added to A
  - Contradiction. Then A is optimal

Your turn: Frog Jumping

- Frog in the river at position 1, want to reach position n.
- Frog can jump at most r units from a position
- Some lilypads at various positions (1, n)
- Find min jumps to go from 0 to n
- E.g. r = 3: lilypads = {1, 3, 4, 6, 8, 9, 11}
  - Your trip can be: {1, 3, 6, 9, 11}

Readings

- Stay Ahead
- Stay Ahead
- Stay ahead with Dijkstra, page 3
- Frog problem

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً