Carlos Mc Gregor c.mcgregormuro@mail.utoronto.ca

CSC373S: Algorithm Design, Analysis & Complexity

Lecture 05

Monday January 16, 2017
based on notes by Denis Pankratov

Greedy Algorithm Paradigm

Make a decision about an input item that looks *best* at the time. Never change your decision!

Activity Selection

Input: Array A of n activities described by:

s_i - starting time

f_i - finishing time

Output: $S \subseteq [n]$ so that activities in S are *compatible* in such a way that:

$$\forall i \neq j \in S [s_i, f_i) \cap [s_j, f_j) = \varnothing$$

and S is as large as possible.

Note: This can have multiple answers.

Trivial solution: consider *all* possible subsets of $[n]$.

Running time: $\Omega(2^n)$ (which is a simplified version of $\mathcal{O}(2^n \text{poly}(n))$)

Algorithm (Generic):

```
def TemplateGreedy(A):
  # assume this runs in O(n log n) (MergeSort)
  sort A accordint to some criterion

  S = []

  # runs in O(n)
  while A != []:
```

```
        # select 1st activity from A
        # add it to S
        # remove all overlapping activities from A

    return S
```

Approaches:

1. Increasing starting time [NOT OPTIMAL]:

$$s_1 \leq s_2 \leq \cdots \leq s_n$$

2. Increasing interval length [NOT OPTIMAL]:

$$f_1 - s_1 \leq f_2 - s_2 \leq \cdots \leq f_n - s_n$$

3. Pick interval that overlaps fewest # of intervals first [NOT OPTIMAL]
4. Earlieset finishing time (EFT) [OPTIMAL]:

$$f_1 \leq f_2 \leq \cdots \leq f_n$$

Definition: Let S_i be the partial solution of EFT prior to i^{th} iteration.
Definition: S_i is feasible if it is possible to expand it to some optimal solution using intervals remaining in A.

Loop Invariant: S_i is feasible.

Proof by induction on i:
Base Case: $i = 0, S_i = \varnothing, A =$ all intervals
Induction Assumption (IA): S_i is feasible for some $i \geq 0$. S_i is extendible to some optimal solution called emphOPT.

$A \neq \varnothing$

Let $a = [s, f)$ be the first element from A:

- Case 1: $a \in OPT$, then we are done.
- Case 2: $a \notin OPT$, let $a' = [s', f')$ be the first interval in OPT, following integers from S_i.
    1. $a \cap a' \neq \varnothing$, otherwise $OPT \cup a$ is a valid solution $> OPT$.
    2. $f \leq f'$ due to algorithm choice.

  $(OPT \setminus a') \cup a$ is another optimal solution that agrees with EFT after i^{th} iteration $\rightarrow S_{i+1}$ is feasible.

Termination: $A = \varnothing \rightarrow S_i$ is optimal
Running Time: $\mathcal{O}(n \log n)$
Notes:

1. Easy to come up with
2. Almost never works: very few problems where it is feasible, so it is advised to always be skeptical and try to prove its wrong!
3. Most useful for approximations (*see later in the course*)

Interval Scheduling to Minimize # of Machines

Input: Array A of n activities, $n \geq 1$
Output: $d \in \mathbb{N}$ so that all activities in A can be scheduled on d machines, but not $d - 1$.

Definition: $\text{depth}(A)$ = max # of intervals passing over a single point on the timeline

Claim: $d \geq \text{depth}(A)$. Surprisingly, $d = \text{depth}(A)$

It is possible to schedule on $\text{depth}(A)$ machines using a *Greedy Algorithm*.

Algorithm:

```
def MinIntervalScheduling(A):
  # sort A by increasing starting time

  init array M of size n

  /* M[i] = # (name) of the machine on which A[i] is scheduled */

  M[1] = 1

  for i = 2 to n:
    S = []

    for j = 1 to i - 1:
      if A[j] intersection A[i] != []
        add M[j] to S

      M[i] = smallest natural # not in S
  return max_{1<=i<=n} M[i]
```