COMPSCI 311: Introduction to Algorithms
Lecture 6: Greedy Algorithms
Dan Sheldon
University of Massachusetts Amherst
11 February 2019

Greedy Algorithms

We are moving on to our study of algorithm design techniques:
▶ Greedy
▶ Divide-and-conquer
▶ Dynamic programming
▶ Network flow

Let’s jump right in, then characterize later what is means to be “greedy”.

Interval Scheduling

▶ In the 80s, you could only watch a given TV show at the time it was broadcast. What if you wanted to watch multiple shows and some of the broadcast times overlap?
▶ You want to watch the highest number of shows. Which subset of shows do you pick?
▶ Fine print: assume you like all shows equally, you only have one TV, and you need to watch shows in their entirety.

Formalizing Interval Scheduling

Let’s formalize the problem
▶ Shows 1, 2, . . . , n (more generally: requests to be fulfilled with a given resource)
▶ $s_j$: start time of show $j$
▶ $f_j$, also written $f(j)$: finish time of show $j$
▶ Shows $i$ and $j$ are compatible if they don’t overlap.
▶ Set $A$ of shows is compatible if all pairs in $A$ are compatible.
▶ Set $A$ of shows is optimal if it is compatible and no other compatible set is larger.

Greedy Algorithms

▶ Main idea in greedy algorithms is to make one choice at a time in a “greedy” fashion.
(Choose the thing that looks best, never look back . . . )
▶ We will sort shows in some “natural order” and choose shows one by one if they’re compatible with the shows already chosen. Concretely:

$$ R \leftarrow \text{set of all shows sorted by some property} $$
$$ A \leftarrow \{ \} $$

▷ selected shows

while $R$ is not empty do
  take first show $i$ from $R$
  add $i$ to $A$
  delete $i$ and all overlapping shows from $R$
end while
4.1 Interval Scheduling: The Greedy Algorithm Stays Ahead

The most obvious rule might be to always select the available request that starts earliest; in (b), it does not work to select the shortest interval; and in (c), it does not work to select the interval with the fewest conflicts.

We'll consider shows in ascending order of finish time. We'll prove that it will always be optimal. Sorting shows by finish time gives an optimal solution in examples. Let's try to prove that it will always be optimal.

Let $A = i_1, \ldots, i_k$ be the intervals selected by the greedy algorithm. Let $O = j_1, \ldots, j_m$ be the intervals of some optimal solution $O$.

Ordering by Finish Time is Optimal: “Greedy Stays Ahead”

- Start Time: Consider shows in ascending order of $s_j$. Not optimal in running example.
- Shortest Time: Consider shows in ascending order of $f_i - s_j$. Not optimal in running example.
- Fewest Conflicts: Let $c_j$ be number of shows which overlap with show $j$. Consider shows in ascending order of $c_j$. Optimal in running example. But not this one:

```
A: |---i1---| |---i2---| ... |---ik---|
O: |---j1---| |---j2---| ... |---jm---|
```

- Finish Time: Consider shows in ascending order of $f_j$. We'll show that this is always optimal!

What's a “natural order”?

- Start Time: Consider shows in ascending order of $s_j$. Not optimal in running example.
- Shortest Time: Consider shows in ascending order of $f_i - s_j$. Not optimal in running example.
- Fewest Conflicts: Let $c_j$ be number of shows which overlap with show $j$. Consider shows in ascending order of $c_j$. Optimal in running example. But not this one:

```
A: |--i1--| ... |---i(r-1)---|
O: |---j1---| ... |---jr-----|
```

- Finish Time: Consider shows in ascending order of $f_j$. We'll show that this is always optimal!
Clicker Question

A: |--i1--||---i2---| ... |---ik---|
O: |---j1---||---j2---| ... |----jm----|

Recall that $k$ is the number of intervals in the greedy solution and $m$ is the number of intervals in an optimal solution. What have we just proven?

A. $f(i_r) \leq f(j_r)$ for $r = 1, 2, \ldots, m$
B. $f(i_r) \leq f(j_r)$ for $r = 1, 2, \ldots, k$
C. The greedy algorithm is optimal.
D. None of the above.

Optimality

A: |--i1--||---i2---| ... |---ik---|
O: |---j1---||---j2---| ... |----jm----|

Can it be the case that $k < m$?

No. Because "greedy stays ahead", intervals $j_{k+1}$ through $j_m$ would be compatible with the greedy solution, and the greedy algorithm would not terminate until adding them.

Running Time?

$R \leftarrow$ set of all shows sorted by finishing time
$A \leftarrow \{\}$

while $R$ is not empty do
  take first show $i$ from $R$
  add $i$ to $A$
  delete $i$ and all overlapping shows from $R$ \(\theta(n^2)\)
end while

Can we make loop better than $n^2$?

Running Time?

$R \leftarrow$ set of all shows sorted by finishing time
$A \leftarrow \{\}$, $\text{end} = 0$ — last scheduled time

for show $i$ from 1 to $n$ do
  if $s_i \geq \text{end}$ then
    add $i$ to $A$, $\text{end} = f_i$ \(\theta(O(1))\)
  end if
end for

$\Theta(n \log n)$ — dominated by sort

Algorithm Design—Greedy

Greedy: make a single "greedy" choice at a time, don’t look back.

Learning goals:

- [ ] Formulate problem
- [ ] Design algorithm
- [X] Prove correctness
- [ ] Analyze running time
- [ ] Specific algorithms — Dijkstra, MST

Focus is on proof techniques. Next time: another proof technique.

Problem 2: Interval Partitioning

- Suppose you are in charge of UMass classrooms.
- There are $n$ classes to be scheduled on a Monday where class $j$ starts at time $s_j$ and finishes at time $f_j$.
- Your goal is to schedule all the classes such that the minimum number of classrooms get used throughout the day. Obviously two classes that overlap can’t use the same room.
Running Example

Greedy Algorithm

- Process classes in order of start time
- For each class, either allocate a new room, or reuse an already allocated room if the last class in that room has completed

```
1 4
2 3
2 7
3 5
4 7
5 7
7 10
```

Clicker Question

If the class with the next starting time is compatible with several rooms, it should be scheduled

A. In the room with the earliest finishing time
B. In the room with the latest finishing time
C. In a room where nothing was scheduled so far
D. Does not matter

Analysis

Let $d$ be the maximum number of classes that run at the same time. Any solution requires at least this many rooms.

**Claim**: the greedy algorithm uses exactly $d$ rooms, and is therefore optimal.

**Proof**: By contradiction.

- Suppose it uses more than $d$ rooms
- There are $d$ classes running when the $d+1$st room is allocated.
- This set of $d+1$ classes overlap, therefore the depth is greater than $d$. 