COMPSCI 330 Lecture 6 Greedy Algorithms

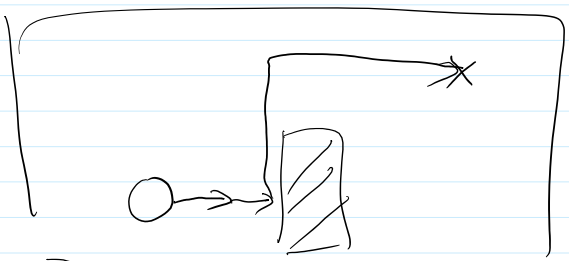
Wednesday, September 14, 2016 1:28 PM

- Greedy Algorithms: General Ideas
- Fractional Knapsack
- Interval Scheduling

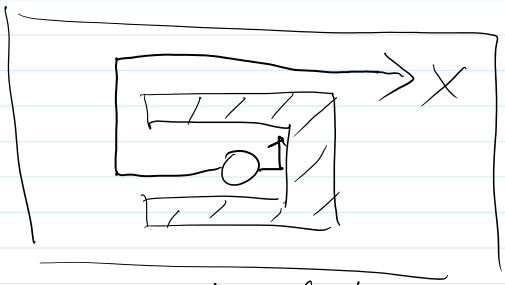
- Basic Idea

  Make decision that looks best now.

- Simple Example: Navigation



Greedy: always move closer to target



greedy fails

- Fractional Knapsack
  - Knapsack with capacity m
  - n items, each item has weight w_i, value v_i
  - items are "dividible" put a fraction of item into knapsack

    can put $\frac{1}{2}$ item 1 weight $\frac{w_1}{2}$ value $\frac{v_1}{2}$
  - Goal: put items into knapsack to get maximum value.

Ex: $m = 10$

$w_1 = 6 \quad v_1 = 20$

$w_2 = 5 \quad v_2 = 15$

$w_3 = 4 \quad v_3 = 10$

Solution: item 1 + $\frac{4}{5}$ · item 2

$$\text{total weight} = 6 + \frac{4}{5} \cdot 5 = 10$$

$$\text{total value} = 20 + \frac{4}{5} \cdot 15 = 32$$

- Q: what is the first item I want to put in

Idea: choose the largest V_i/W_i

- Algorithm

sort items in decreasing order of V_i/W_i

while knapsack is not full

put the item with largest V_i/W_i remaining into knapsack

- Analysis: General Idea: Proof by Contradiction

(1) assume the optimal solution is different

(2a) show that we can improve optimal solution

Proof: First, if two items have same ratio V_i/W_i

$V_1/W_1 = V_2/W_2$, then can combine these items

($V_1 = 20$, $W_1 = 6$ $V_2 = 10$ $W_2 = 3 \Rightarrow V = 30$ $W = 9$)

Without loss of generality can assume $V_1/W_1 > V_2/W_2 > V_3/W_3 > \cdots > V_n/W_n > 0$

(1) { algorithm select weights $P_1, P_2, \cdots, P_n$ (6 4 0)

assume optimal solution $q_1, q_2, \cdots, q_n \neq P_1, P_2, \cdots, P_n$ (6 3 1)

(2) Let i be the first index where $P_i \neq q_i$

by design of algorithm $P_i > q_i$

if we increase q_i to P_i, decrease some q_j's $(j > i)$ (maintain $\sum_{i=1}^{n} q_i$)

value of optimal solution increases!

but this contradicts with the assumption $q_1, q_2, \cdots, q_n$ is optimal

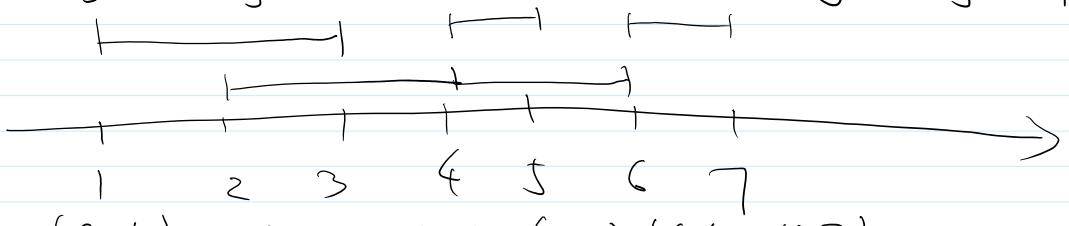
Note: greedy does not work for integer knapsack. □

- Example: Interval Scheduling

- Trying to schedule meetings. There are n requests

each request has starting time S_i and ending time t_i

- Design an algorithm that can schedule as many meetings as possible.



$(s_i, t_i) = (1,3), (2,4), (4,5), (4,6), (6,7)$

optimal: 3 meetings $(1,3)$ $(4,5)$ $(6,7)$

$((2,4)$ $(4,6)$ $(6,7))$

Q: what is the first meeting to schedule?

A: the one with earliest end time.

ALG: Sort meetings in increasing order of end time

for $i = 1$ to n

if meeting i can be scheduled

Put meeting i on schedule

$(1,3)$ ~~$(2,4)$~~ $(4,5)$ ~~$(4,6)$~~ $(6,7)$

Proof: assume the meetings alg scheduled is

$u_1, u_2, \cdots, u_k$ $(1, 3, 5)$ $(1,3)$ $(4,5)$ $(6,7)$

assume optimal solution is

$v_1, v_2, \cdots, v_l$ $l > k$ (OPT > ALG)

(will show this is impossible)

(always think of $(u_1, \cdots, u_k)$, $(v_1, \cdots v_l)$ are sorted in time)

Claim: If i $(1 \le i \le k)$ is the smallest index where $u_i \neq v_i$

u_1 u_2 $\cdots$ u_{i-1} are the same

v_1 v_2 $\cdots$ v_{i-1}

$(u_1, u_2, \cdots, u_i, v_{i+1}, \cdots, v_l)$ is also a solution)

by design of algorithm: u_i is the meeting with earliest end time, among meetings compatible with $u_1, u_2, \cdots, u_{i-1}$

in particular end time of $u_i \le$ end time of v_i

$(u_1, u_2, \cdots, u_i, v_{i+1}, \cdots, v_l)$ is also a solution, because

start time of $v_{i+1} \ge$ end time of v_i

$\geq$ end time of u_i

Keep modifying optimal solution by Claim 1, then get an optimal solution where $u_i = v_i$ for all $i = 1, 2, \cdots, k$

alg u_1 u_2 $\cdots$ u_k

opt u_1 u_2 $\cdots$ u_k v_{k+1} $\cdots$ v_ℓ

by design of alg, if $\exists$ a meeting compatible with $u_1, u_2, \cdots, u_k$ it will be scheduled.

v_{k+1} cannot exist.

This contradicts with our assumption.

□