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

CSC373S: Algorithm Design, Analysis & Complexity

Tutorial 02

Monday January 23, 2017

based on notes by TA (LM 161 - Colin Li qiyang.li@mail.utoronto.ca)

Greedy Algorithm

16-1

Input: m types of coins, $v_1, v_2, \ldots, v_m$, 1-cent is a penny, 5-cent is a nickel, 10-cent is a dime, 25-cent is a quarter
Output: minimum # of coins n cents

Example:
n cents:
1 25-cent $\rightarrow$ 32 - 25 = 7
1 5-cent $\rightarrow$ 7 - 5 = 2
2 1-cent $\rightarrow$ 2 - 2 = 0

making change for n cents $\rightarrow$ pick a coin c $\rightarrow$ making change for $n - c$ cents
This can be seen as:
Optimal Solution $\rightarrow$ Coin c lies in the optimal solution of 'making change for n-cent' $\rightarrow$ Optimal Solution

We apply a *Recursive Greedy Solution*:

Case 1: $1 \leq n < 5$
Greedy: 1-cent
making change for n-cents $\rightarrow$ $(n-1)$ cents
Greedy Solution is Optimal

Case 2: $5 \leq n < 10$
Greedy: 5-cent
$n \rightarrow n - 5$

Case 3: $10 \leq n < 25$
Greedy: 10-cent
$n \rightarrow n - 10$
if 10-cent is not in the optimal solution, replace pennies (1-cent) and nickels (5-cent) by a dime (10-cent)

Case 4: $n \geq 25$
Greedy: 25-cent
Greedy Solution is Optimal

b) $m = k+1$, $v_1 = c^0, v_2 = c^1, \ldots, v_m = c^k; k, c \in \mathbb{N}^+$
Let a_i be the # of c^i we use in an optimal solution:

$$a_i < c, i < k$$

Greedy Choice: if $c^i \leq< c^{i+1}, i < k$, we pick $c^i, n \to n - c^i$
if $a_i \geq c$, replace $c \cdot c^i$ by $1 \cdot c^{i+1}$

<u>Runtime:</u> $\mathcal{O}(m) = \mathcal{O}(\log n)$

Case i:

$$c^i \leq n < c^{i+1}$$

$$n \to n - c^i, a_i < c$$

Assume c^i is not in the optimal solution

$$a_i = 0$$

$$\begin{aligned} n &= \sum_{j=0}^{k} a_j c^j \\ &= \sum_{j=0}^{i-1} a_j c^j \\ &= a_0 c^0 + a_1 c^1 + \cdots + a_{i-1} c^{i-1} \\ &< c \cdot c^0 + a_1 c^1 + \cdots + a_{i-1} c^{i-1} \\ &= c(a_1 + 1) + \cdots + a_{i-1} c^{i-1} \end{aligned}$$

$$a_i < c \to a_i + 1 \leq c$$

$$\begin{aligned} (a_1+1)c + a_2c^2 + \cdots + a_{i-1}c^{i-1} &\leq c^2 + a_2 c^2 + \cdots + a_{i-1} c^{2-1} \\ &= (a_2+1)c^2 + \cdots + a_{i-1}c^{i-1} \\ &\vdots \qquad\qquad \leq c^i \end{aligned}$$

$$n < c^i \to \text{contradicting} \to c^i \leq n < c^{i+1}$$

$a_i = 0$, which is not true. Hence, c^i is in the optimal solution. *Case k:* $n \geq c^k$

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

16-2

$S = (a_1, \dots, a_n)$ **Input:** p_i processing time for task a_i. Computer can only perform one task at a time. c_i complete time for task a_i.
Output: minimize $C_{avg} = {}^1\!/\!_n \sum_{i=1}^{n} C_i$

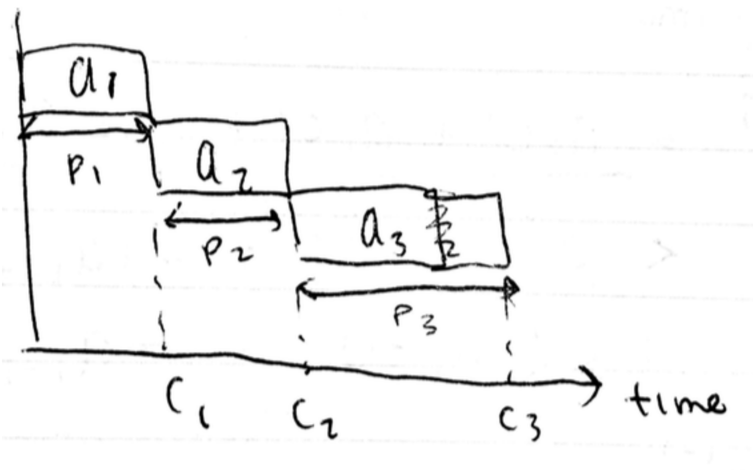

Figure 1: Visualization of problem (*courtesy of Vicky Lu*)

This can be solved using the *Greedy Algorithm*:

$$c_i = p_1 + p_2 + \cdots + p_i$$

$$\begin{aligned} C_{avg} &= {}^1\!/\!_n[p_1 + (p_1 + p_2) + \cdots + (p_1 + \cdots + p_n)] \\ &= {}^1\!/\!_n[n \cdot p_1 + (n-1)p_2 + \cdots + 2p_{n-1} + p_n] \end{aligned}$$

Algorithm (General Idea):

```
def Greedy(p):
  # attempt to do p[1] <= p[2] <= ... <= p[n]
   if p[i] > p[j], i < j:
     # if we find that something is not order, we can always swap
     # it to get a better join
     swap tasks i, j

  # check if our new average is better
  c_avg < c_avg
```