Analysis of Algorithms

Greedy Algorithms

Greedy Algorithm

- Class of algorithms whose choices are locally best.
- Choices are greedy: “live for today” motto.
- Often very simple and efficient algorithms, but not always possible for all problems.
- Example: Coin Changing
  - 4 types of coins: (25, 10, 5, 1)
  - Given integer x between 9 and 99, make change for x with least number of coins.

Greedy Algorithms

- Mathematically, write

  x = 25 a + 10 b +5 c+ 1d so that a + b + c + d is minimum and a,b,c,d >= 0 are integers.
- Suggest an algorithm for the coin changing problem.

Greedy Coin Changing

- Choose as many quarters as possible.
- That is , find largest a so that 25a <=x.
- Next, choose as man dimes as possible to change x – 25a, and so on.

Greedy Coin Changing

- An Example. Consider x =73
  - Choose 2 quarters, so a = 2.
  - Left 73 – 2 x 25 = 23
  - Next Choose 2 dimes, so b = 2. Left 23 – 2 x 10 = 3.
  - Choose 0 nickels, so c = 0. Left 3
  - Finally, choose 3 pennies, so d =3. Left 3 – 3 =0.
  - Solution a = 2, b = 2, c = 0, d = 3.

When Greedy Fails

- Greedy algorithms correctness depends on the choice of coins.
- When coins have denominations (25, 10, 5, 1), the greedy always works for any x.
- But consider the case when coins are of types (12, 5, 1)
- The greedy does not always return the optimal solution.

When Greedy Fails

- For x = 15, the greedy will use 4 coins: 1 * 12 + 0 * 5 + 3 * 1.
- The optimal uses 3 coins: 3* 5
- Moral: Greed, the quick path to success or to ruin!.

Related Problems

- Frobenious Problem: Given coins of denomination d1, d2, …,dn, so that no two have a common factor, find the smallest integer x that cannot be changed using these coins.
- Frobenius problem sis NP- Hard!.

Activity Selection

- Scheduling a resource among competing activities; e.g. processor and jobs, a lecture hall and academic activities etc.
- When an activity is scheduled, it must be run to completion; no pre-emption.
- Let S = {1,2,….,n} be a set of activities that demand the use of a lecture hall.

Activity Selection

- The activity i has a start time s_i and a finish time f_i . Obviously, $s_i <= f_i$. The hall can be used by at most b one activity at any time moment.
- We want to schedule as many activities as possible, subject to single use constraint.
- Suggest an algorithm to solve this problem.

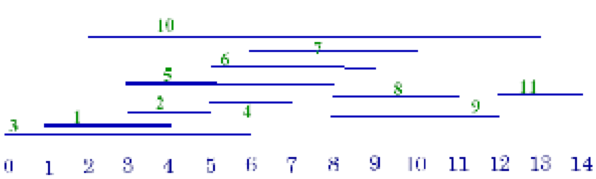
Activity Selection

- Example: Given the starting time s_i and finish time f_i of each activity, where $0 \le s_i < f_i < \infty$:

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

Activity Selection

- Consider the example of 11 activities, shown below.



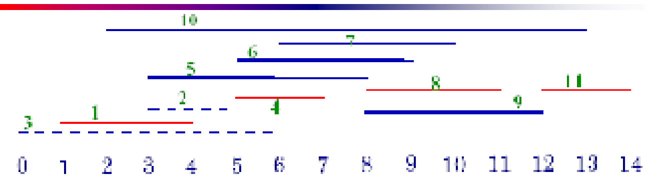
Greedy Selection Algorithm

- Sort the activities in increasing finish time order. Takes O(n log n) time.
- So, assume that $f_1 \le f_2 \le \ldots. \le f_n$.
- Scan the activities in this order and choose the first one that doesn't conflict with already chosen activities. Repeat, until we reach the end of the activity list.

Greedy Selection Algorithm

```
GREEDY-ACTIVITY-SELECTOR(s, f)
n ← length[s]
A ← {a1}
i ← 1
for m ← 2 to n
  do if s_m ≥ f_i
       then A ← A U {a_m}
       i ← m
return A
```


Example



- The greedy algorithm first chooses 1; then skips 2 and 3; next it chooses 4, and skips 5, 6, 7; so on.