P-①

Homework - 4 Amit Pandit
CS-575 apandit1@binghamton.edu

Q1]

→ i) Activity selection of least duration will not work.
Example:- if we consider these are 4 activities
1 to 4 which has different duration for each activity start time and finish time is depend as below

Activity	start time	finish time	Duration
1	3	5	2
2	4	8	4
3	1	4	3
4	1	5	4

solution for least duration activity is {1} that is 1st activity is duration is 2 but optimal solution is {2,3} that is activity 2 and 3

ii) Always selecting the compatible activity that overlaps the fewest other remaining activities this will not work, consider below example as there are 6 different activities each activity has start and finish time. these activities may or may not overlap with other activities start and finish time is defined in below table also overlap count will be added with other activities

P-6

Activities	Start time	Finish time	overlap
1	2	4	3
2	3	5	3
3	4	7	2
4	1	3	2
5	2	4	3
6	6	8	2
7	7	10	3
8	9	11	2
9	7	10	3

Solution for above approach is {3, 4, 8} or {4, 6, 8}

but optimal solution is {2, 4, 6, 8}

iii) Selecting the compatible remaining activity with earliest start time. This will not work Consider below example.

Activity	Start time	Finish time	Duration
1	1	10	9
2	2	4	2
3	5	7	2
4	4	10	3

solution for earliest start time is {1}

but optimal solution is {2, 3, 4}

Q2) if we consider problem of making change for n cents using fewer number of coins

a) penny - 1
Nickle - 5
Dime - 10
quarter - 25

Consider 'n' be the no of cents. Now first devide the number of cents (n) with quarters (a) the floor value will be no of quarters if reminder is greater than 0 then reminder value to be devide by dime here floor value will be number of dime if reminder is greater than 0 then again devide reminder with Nickle if floor value is zero then that will be number of Nickle & if reminder is non zero then we will ~~devide reminder with~~ consider reminder as no of ~~Cents~~ penny

Number of quarter = (n/25)
n = n mod 25
if n (n mod 25) is greater than 0 Then
no dime = (n/10)
n = n mod 10
if n (n mod 10) is greater than 0 then
no of Nickles = (n/5)
n = n mod 5

P-(5)

~~if n(n mod 5) is gre~~

Number of pennies = p = n

Here we have 5 denominations

- if we have n = 52, then there are diffrent soln consider 52 pennies or 5 dime 2 penny etc but optimal solution will be 2 quaters & 2 penny
- if we have n = 23 then are more than 1 solution as we can consider 13 pennies or 2 dime 3 pennies but optimal solution will be 1 dime 3 pennies

hence there is always a greedy optimal solution for every value of n

b)

if available coins are in the denomination that are power of c,

denominator are $c^0, c^1, \ldots, c^k$

for integers $c > 1$ and $k > 1$

no of cents = n

$c > 1$ and $k \geq 1$

then $c^0, c^1, c^2 \ldots, c^k$ are denominators

maximum value for denominators has to be less than value of n, we have to make

Change for number of cents. we can calculate value of k starting from 0 to c' (i=0; i<=c'; i++) & comparing each c' value with n if c' is > n then we should break the loop the i-1 = k'

Now we have traverse from $c^0, c^1, \ldots c^k$ from the

Number of Coins for $c^k = \left[\frac{c^k}{n}\right]$

Number of coins for $c^{k-1} = \left[\frac{c^k \bmod c^{k-1}}{c^{k-1}}\right]$

repeat until k=0

after k=0 we will get number of coins required to make change of n cents

Eg. if n=25 cents & c=3

then max denomination is 3^2

3^3 is 27 which greater than quater to k=2

$3^0 = 1$

$3^1 = 3$

$3^2 = 9$

The optimal solution is $3^2-2, 3^1-2$ & 3^0-1 algorithm also give $\{3^0-25\}$ or $\{3^1-6, 3^2-7\}$ etc which are not optimal solution

p-⑥

c]

for example denominations are = {1, 5, 8} and cents = 20

so n = 20

greedy algorithm give {4, 0, 2} coins respectively above solutn give 4+2=6 coins but optimal solⁿ is {0, 4, 0} only 4 coins can give end result instead of 6 coins

d]

```
PowerChange (int cents, int den_arr[], int size)
    int num_unit;
    for (i=size, i>=0 && cents>0; i--)
    {
      if (den_arr[i] < cents)
       {
            num_unit = cents / den_arr[i];
       cents = cents - num_unit * den_arr[i];
      print ("Denominations %d : den_arr[i];
       }
    }
```

den_arey is containing denominations wher one coin is penny & size is length of denominations

above algorithm runs in O(k) time
duration takes (O(1)) time