ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA FALL 2015

CWID: A20355611

1.(a) Consider the set of coin ~~val~~ values mentioned below.

$n=6$ and values are $\{1, 3, 6, 10, 20, 50\}$

Now, if we want to give a change ~~for~~ of 18 cents. Applying the greedy algorithm, we need to give a total of 4 coins. ~~(2*1+~~ $((1 * 10)+(1*6)+(2*1))$, where as the optimal solution has only 3 coins $(3*6)$.

ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA FALL 2015

CWID: A20355611

1.(b) In this problem the coin denominations are as follows $b^0, b^1, b^2 \ldots b^k$ ~~of~~ for $b \geq 2$.

- The greedy algorithm ~~states~~ asks us to use the highest possible coins ~~on~~ to give the change. Lets say 'S' is the amount for the change to be given. Lets choose a particular denomination 'b^i' and we express 'S' as follows

$$S = \sum_{n=i+1}^{k} f_n \cdot b^n + f_i \cdot b^i + \sum_{m=0}^{i-1} f_m \cdot b^m$$

(i) Lets assume that, we have made ~~a~~ our first mistake by choosing 'b^i' denomination. Now, we ~~need~~ choose another denomination 'b^j'.

$$S = \sum_{n=i+1}^{k} f_n \cdot b^n + f_j \cdot b^j + \sum_{m=0}^{j} f_m \cdot b^m$$

Note that the first part did not change because, that is already optimal solution. ~~Hen~~

Now, we know for sure that $b^i > b^j$, and even if we assume that the second part ~~$\sum_m$~~ of i.e. $\sum_{m=0}^{i} f_m \cdot b^m$ and

ASSIGNMENT : HW 2

NAME : PRASANTH BHAGAVATULA

FALL 2015

CWID : A 20355611

$\sum_{m=0}^{j} f_m \cdot b^m$ both are same ~~with~~ then $f_i < f_j$ inorder to make the values same. Hence the number of coins when 'b^i' used is less than the number of coins when 'b^j'

Infact, for any '$m < i$'; $b^i = b^{i-m} \cdot b^m$ and so the number of coins of denominations of (b^m) is (b^{i-m}) th times the number of coins of denomination of (b^i). Hence, it is always optimal to use the highest possible denomination and so the greedy algorithm gives an optimal solution.

ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA

FALL 2015

CWID: A20355611

1.. (c) Consider the amount as A and sorted array of coin denominations $C[1 \cdots n]$. Now, the function $f(S, C[1 \cdots k])$ gives us the smallest number of coins needed to make amount S in cents using denominations in $C[1 \cdots k]$.

Consider the highest denomination C_k; Now there can be two possibilities that the coin might be present in the optimal set (or) it might not be. So, we can express the functions as

$$f(S, C[1 \cdots k]) = \min\left(1 + f((S - C_k), C[1 \cdots k]),\ f(S, C[1 \cdots k-1])\right)$$

The first possibility is that one coin of denomination C_k is considered and hence, we need to find the optimal number of coins for the remaining amount $(S - C_k)$.

The second possibility is that C_k is not at all present, hence we consider only the remaining coin denominations $C[1 \cdots k-1]$ for the amount S.

ASSIGNMENT : HW2

NAME : PRAJANTH BHAGAVATULA

CWID : A20355611

FALL 2015

Now, the breaking condition would be when the amount 'S' is equal to '0'. In this case, we require no (or) '0' coins and hence we return '0'. So, the final expression is

$$f(S, C[1 \cdots k]) = \begin{cases} \min\left(\begin{matrix} 1 + f((S-C_k), C[1 \cdots k]), \\ f(S, C[1 \cdots k-1]) \end{matrix} \right), & \text{if } S > 0 \\ 0, & \text{if } S = 0 \end{cases}$$

Below is the algorithm.

$f(A, C[1 \cdots n])$

```
If A == 0
    Return 0
Else
    If ((A - C_n) >= 0 and (n-1) > 0)
        Return min(1 + f((A - C_n), C[1....n]), f(A, C[1....n-1]))
    Else
        If (A - C_n) >= 0
            Return 1 + f((A - C_n), C[1....n])
        Else
            Return f(A, C[1....n-1])
```

ASSIGNMENT : HW2

NAME: PRASANTH BHAGAVATULA FALL 2015

CWID: A20355611

Note that the sum $(A - C_n)$ can never be less than zero and $(n-1)$ should be greater than zero (at least we should have one denomination). Hence the IF-ELSE conditions.

For a given set of denominations, ~~these~~ is the optimal solution (minimum number of coins) to get sum S is always fixed. and here in the algorithm, we are calling recursively for the same amount even after the optimal solution is determined. So, we add ~~the~~ memoization to this algorithm.

$f(A, C[1 \cdots n])$

```
If A == 0
    Return 0
Else
    If M[A, n] exists
        Return M[A, n]
    Else
        If (A - Cn) >= 0 and (n-1) > 0
            M[A, n] = min(1 + f((A-Cn), C[1...n]), f(A, C[1....n-1]))
        Else
            If (A - Cn) >= 0
                M[A, n] = 1 + f((A-Cn), C[1...n])
            Else
                M(A, n) = f(A, C[1....n-1])
    Return M[A, n]
```

ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA

FALL 2015

CWID: A20355611

The ~~m~~ 2-dimentional array M[A, n] represents the minimum number of coins required to get a sum of 'A' ~~in~~ using a set of 'n' denomination. So, the algorithm needs time just to fill up the 2-dimentional array and so the asymptotic time ~~for~~ this algorithm would be $\theta(A * n)$.

ASSIGNMENT. HW2

NAME: PRASANTH BHAGAVATULA

FALL 2015

CWID: A20355611

2. (i) Argue that the optimal number of bins required is atleast $\lceil S \rceil$

Ans. Given that $S = \sum_{i=1}^{n} s_i$. Now, we can choose an object s_k such that all the objects from $s_1, s_2 \cdots s_k$, all fit into 1 bin.

So, $\sum_{i=1}^{k} s_i \leq 1$. Now we can express S as

$$S = \sum_{i=1}^{n} s_i$$

$$\Rightarrow S = \sum_{i=1}^{k} s_i + \sum_{j=k+1}^{n} s_j.$$

$$\Rightarrow S \leq 1 + \sum_{j=k+1}^{n} s_j$$

Applying the same logic again and again for the remaining elements, we will finally reach to an integer which is greater than S and the first integer which is greater than S is $\lceil S \rceil$.

So, we need atleast $\lceil S \rceil$ bins. to fit all the given 'n' objects.

ASSIGNMENT: HW2
NAME: PRASANTH BHAGAVATULA
CWID: A20355611

FALL 2015

2 (ii) Argue that first-fit heuristic leaves at-most one bin less than half full.

Ans:

Lets assume that 'i' objects are present in a bin and that bin is less than half full. If we assume that the next object, $(i+1)^{th}$ object, is less than of bin half ~~full~~ and it is placed in another bin (so that 2 bins are there with less than half full). But, as per the algorithm since the object can be fitted into the current bin (i)' it will not open a new-bin. ~~Hence~~ there are ~~no~~ bins So, the $(i+1)^{th}$ object will go into the last bin. Now there can be two cases, either the bin can be more than half ~~full~~ full (or) at most half full. In any case, ~~only~~ at most ~~max~~, only one bin is half full.

Now, if we assume that the $(i+1)^{th}$ object is more than half full, ~~th~~ and it cannot be accommodated in the last bin, then a new bin will be opened. But since this object is more than half full, the last bin will be more than half full but the last but one bin is ~~be~~ less than half full

ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA

CWID: A20355611

FALL 2015

2. (iii) Prove that the number of bins used by the first-fit heuristic is never more than $\lceil 2S \rceil$

Ans. Consider, a scenario where in each object is just above half of the size of the bin ie $s_i > 1/2$, so that no two objects can fit together into a single bin

$$s_i > 1/2$$

$$\Rightarrow \sum_{i=1}^{n} s_i > \sum_{i=1}^{n} 1/2$$

$$\Rightarrow S > n/2$$

$$\Rightarrow 2S > n$$

$$\Rightarrow \lceil 2S \rceil > n$$

Since no two objects can be fit together into a single bin and there are 'n' such objects; we require 'n' bins which is never more than $\lceil 2S \rceil$ bins.

ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA

FALL 2015

CWID: A20355611

2. (iv) Prove an approximation ratio of 2 for the first-fit heuristic

Ans. From the previous question, we have proved that a ~~maximum~~ the number of bins can never be more than $\lceil 2S \rceil$.

~~Consider~~ Let 'k' be the number of bins.

$$k < \lceil 2S \rceil$$

We know for sure that $\lceil 2S \rceil \leq 2\lceil S \rceil$. Consider S as 1.4, then $\lceil 2S \rceil = 3$ and $2\lceil S \rceil = 4$. If $S = 1.6$, then $\lceil 2S \rceil = 4$ and $2\lceil S \rceil = 4$.

So,

$$k < \lceil 2S \rceil \leq 2\lceil S \rceil$$

$$\Rightarrow k < 2\lceil S \rceil$$

$$\Rightarrow k < 2\,(\text{OPTIMUM\_VALUE})$$ (proved in the first question)

(*) Alternatively, we ~~e~~ from the second question, we can say that at most one bin is ~~half~~ less than half full.

Meaning, ~~all~~ all other bins are more than half full.

(*) This proof is in reference to the page present in wikipedia

ASSIGNMENT: HW2

NAME: PRASANTH BHAGAVATULA

CWID: A20355611

FALL 2015

Hence, we can say that

$$\frac{(k-1)}{2} < \sum_{i=1}^{n} s_i$$

$(k-1)$ bins are more than half filled and so, $\frac{(k-1)}{2}$ is the half of size of $(k-1)$ bins which is less than the size of 'n' objects

From the first question we have proved that the optimum number of bins would be $\lceil s \rceil$.

So,

$$\frac{(k-1)}{2} < \lceil s \rceil$$

$$\Rightarrow \quad k-1 < 2\lceil s \rceil$$

$$\Rightarrow \quad k \leq 2\lceil s \rceil$$

Hence the number of bins required by the ~~approximation~~ first-fit heuristic algorithm has an approximation ~~of~~ factor of 2 with respect to optimal number of bins.