Anuj Panwar
Varun Goyal

Solution 1.

Pseudo-code:

```
SELECT-RAND (A, k)   //N is length of array A
 1.   x = A[i]  where i = a random number from {1,…,n}
 2.   rearrange A so that all elements smaller than
      x come before x, all elements larger than x
      come after x, and elements equal to x are
      next to x
 3.    j1,j2 = the leftmost and the rightmost
       position of x in rearranged A
 4.   if (k<j1) return SELECT-RAND(A[1…(j1-1)],k)
 5.        if (j1<=k<=j2) return x
 6.    if (k>j2) return SELECT-RAND(A[(j2+1)…n],k-j)
```

```
// X is the array with coordinates (x1,x2,x3……..xn)
1. Float Xbest = SELECT-RAND(X, n/2)
2. Calculate the sum of distance between each house and Xbest
3. Print sum.
```

Running Time:

O(n)

Argument:

We need $dist_{best} = {}^{n}\sum_{i=1} |x_{best} - x_i|$ to be minimum.

For $dist_{best}$ to be minimum differentiation ($dist_{best}$/ x_{best}) = 0 …. As the function can take as high value as we want by taking x_{best} very large.

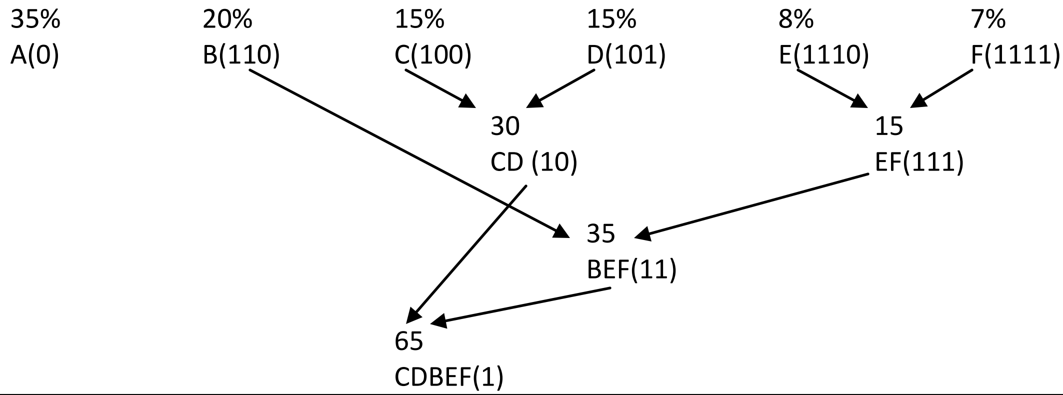
Thus,
$d\, dist_{best} / d\, x_{best} = {}^{n}\sum_{i=1} (x_{best} - x_i) / |x_{best} - x_i| = 0$…………. $d/dx\, |x| = x/|x|$

$= (x_{best} - x_1) / |x_{best} - x_1| + (x_{best} - x_2) / |x_{best} - x_2| + (x_{best} - x_3) / |x_{best} - x_3| + \ldots\ldots\ldots (x_{best} - x_n) / |x_{best} - x_n| = 0$

To make it zero we need as many co-ordinates after x_{best} as many before it. Then only the summation can add up to zero.
Thus, x_{best} should be such that number of co-ordinates before and after it should be equal.
Thus, n is odd then x_{best} should be the middle element, and if n is even, then x_{best} can take any position between the two center most positions.

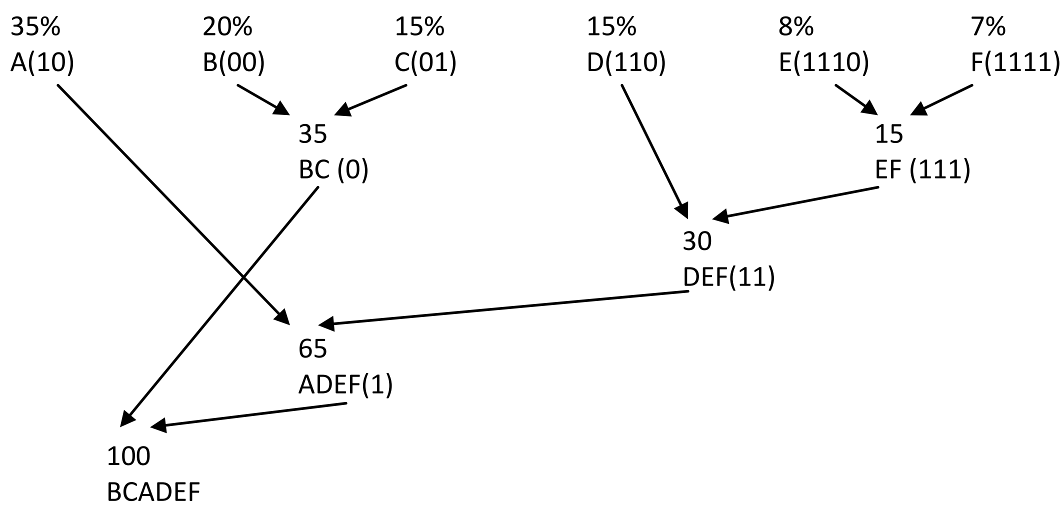
Anuj Panwar
Varun Goyal

Solution 2.



Character	Code	Bits	Frequency
A	0	1	0.35
B	110	3	0.20
C	100	3	0.15
D	101	3	0.15
E	1110	4	0.08
F	1111	4	0.07

Expected Code-word Length = (1*0.35) + (3*0.20) + (3*0.15) + (3*0.15) + (4*0.08) + (4*0.07) = **2.45**



Character	Code	Bits	Frequency
A	10	2	0.35
B	00	2	0.20
C	01	2	0.15
D	110	3	0.15
E	1110	4	0.08
F	1111	4	0.07

Expected Code-word Length = (2*0.35) + (2*0.20) + (2*0.15) + (3*0.15) + (4*0.08) + (4*0.07) = **2.45**

Anuj Panwar
Varun Goyal

Solution 3.

Observation about part1

The KNAP-IND-REC Algorithm only provides us with the best cost and not the set of items that should be stolen! Also it takes exponential time, thus the program "virtually" hangs when a list of only 50 items is provided to it.
On the other hand KNAPSACK-INDIVISIBLE gives us the best cost very quickly. In time: O(n.W)

Part 2 Pseudo-Code

```
KNAPSACK-INDIVISIBLE(n,c,w,W)
        1. initialize S[0][v]=0 for every v=0,…,W
        2. initialize S[k][0]=0 for every k=0,…,n
        3. for v=1 to W do
                4. for k=1 to n do
                        5. S[k][v] = S[k-1][v]
                        6. if (w_k ≤ v) AND (S[k-1][v-w_k]+c_k > S[k][v]) Then,
                                7. S[k][v] = S[k-1][v-w_k] + c_k
        8. print S[n][W]            //max. cost.
        9. capacityLeft = j = W;
        10. for i=n to 1 do
                11. if (j>0  AND S[i][j]!=S[i-1][j])
                                12. temp[z] = i;
                                13. z++;
                                14. j = capacityLeft - w[i];
                        15. capacityLeft = j;
                16. print temp[] in reverse order.          // To get the items to be stolen!
```

Explanation:

Till point 8 it's same as **KNAPSACK – indivisible: a dyn-prog** as in the professor Bezáková's notes.

1) Once we have the array with max – cost we pick the bottom right element (ie the max cost)
2) and check if it is same as the element above it.
   i) If it is then we pick the element above it. And goto step 2.
   ii) If NOT then we store the row number in the temp.
      a. And go back to the column subtracting the weight of the item stored in temp from total weight W.
      b. Goto step 2.