CS-430 Assignment-6

① A location Sn can be included in the list of locations if the distance between n and previous included location is greater than k. Profit at a location if distance is greater than k will be maximum of profit we can get by including/excluding the location.

Algorithm:

```
Profit (d[], i, n)
  {
   if (n < 0)
       return 0
   else if (i>0 && n-i < k)
       return profit (d[], i, n-1)
   else
       return max [d(n) + profit (d[], i, n-1),
                  profit (d[n]+d[], n, n-1]
  }
```

② Let the array P[n] → holds list of patent
m[n] → holds list of cost
V[n] → holds list of values
M → allocated budget

Algorithm:

```
for m = 0 to M
    P[0, m] = 0
for i = 1 to n
    P[i, 0] = 0
for i = 1 to n
    for m = 0 to M
        if m_i <= M   // i can be a part of solution
            if v_i + P[i-1, m-m_i] > P[i-1, m]
                P[i, m] = v_i + P[i-1, m-m_i]
            else
                P[i, m] = P[i-1, m]
        else
            P[i, m] = P[i-1, m]   // m_i > m
```

P[n, M] → returns the maximum total value that DoNoEvil company can buy within its budget.

TO FIND THE PATENTS THAT CONTRIBUTE TO MAXIMUM VALUE

assume i = n and k = M from P[n, M]

while $i, K > 0$

  if $P[i, K] \neq P[i-1, K]$; ~~return i~~

    $i = i-1$, $K = K - w$; // mark the i^{th}

   Patents[n] ← P_i patent as included

  else

    $i = i-1$ // i^{th} patent is not included.

 ~~return~~

 return patents [n]

Patents array will hold the list of patents with maximum total value within company's budget

③ <u>Maximum sized subsequence</u>:

Given string of characters: $X[0 \ldots n-1]$.

$MSS[0 \ldots n-1]$ → Maximum size subsequence which is a palindrome.

Compare first (i) and last (j) characters of the sequence.

If (i == j)

 → add 2 to result. Remove 2 character and solve for remaining ones.

else

 → find Max $[R(i, j-1), R(i+1, j)]$

Conditions

~~for~~ ① R(i, j) if j = i
→ returns 1

② R(i, j) if j = i + 1
→ returns 2

③ R(i, j) if first and last characters are same
→ return 2 + R(i+1, j-1)

④ R(i, j) if first and last characters are different
→ return Max [R(i+1, j), R(i, j-1)]

Algorithm:

```
for i = 0 to n
    table [i] [i] = 1
for k = 2 to n+1
    for i = 0 to n-k+1
        j = i+k-1
        if (i = = j and k==2)
            table [i] [j] = 2
        else if (i = = j)
            table [i][j] = 2 + table [i+1] [j-1]
        else
            table [i] [j] = Max [(table [i] [j-1],
                                  table [i+1][j ])]
return table [0] [n-1];
```

④ Bitonic Euclidean TSP:

1. Enumerate the points from left to right after sorting.

$B[i, k]$ → Minimum length of 2 disjoint bitonic paths (i to 1 to k)

i) if $(i = k)$ → Min. cost bitonic tour through first i points.

ii) if $(i = k = n)$ → Min. cost bitonic tour thro. first n points.

$$B[n, n] = B[n-1, n] + |P_{n-1} P_n|$$

$$B[i, j] = B[i, j-1] + |P_{j-1} P_j| \quad \text{if } i < j-1$$

$$B[j-1, j] = \min_{1 \leq k < j-1} \left\{ B[k, j-1] + |P_k P_j| \right.$$

Any Bitonic path ending at P_2 has P_2 as it rightmost point, so it consists only of P_1 and P_2. Its length is $|P_1 P_2|$

Consider a shortest bitonic path P_{ij}.
If P_{j-1} is on rightgoing subpath, then it immediately precedes P_j

Length of $P_{ij} \rightarrow b[i, j-1] + |P_{j-1}, P_j|$

sort with respect to x coordinates

Euclidean-TSP(p)

$b[1,2] \leftarrow |P_1 P_2|$
for $j = 3$ to n
    for $i = 1$ to $j-2$
        $b[i,j] = b[i,j-1] + |P_{j-1} P_j|$
        $r[i,j] = j-1$
        $b[j-1, j] \leftarrow \infty$
        for $k = 1$ to $j-2$
            $q = b[k, j-1] + |P_k P_j|$
            if $q < b[j-1, j]$
                $b[j-1, j] = q$
                $r[j-1, j] = k$
            $b[n, n] = b[n-1, n] + |P_{n-1} P_n|$

Print-tour (r, n)
    print P_n
    Print P_{n-1}
    $k = r[n-1, n]$
    print-path (r, k, n-1)
        print P_k

```
print-path (r,i,j)
  if i<j
     k = r[i,j]
     print P_k
  if k ≥ 1
     print-path (r,i,k)
  else
     k = r[j,i]
     if k ≥ 1
        print_path (r,k,j)
        print P_k
```

Time to run Euclidean TSP $\rightarrow O(n^2)$.