

Qn2,

A,

Let $X = \{x_i, x_{i+1}, \dots, x_j\}$, we sort and re-label the points so that

$$x_i \leq x_{i+1} \leq \dots \leq x_j.$$

Let S_{ij} be the smallest set of intervals we build (the solution), contains some intervals $I_k = [x_k, x_k + 1]$, where $i \leq k \leq j$.

Same definition for S_{jk} and S_{jk} .

$$X_{ik} = S_{jk} \cap X$$

$$X_{kj} = S_{jk} \cap X$$

1, S_{ik} is the optimal solution for sub-problem X_{ik}

2, S_{kj} is the optimal solution for sub-problem X_{kj}

B,

Consider any nonempty sub-problem X_k , and let x_1 be a point in X_k with the smallest position. Then interval $I_1 = [x_1, x_1+1]$ is included in some of smallest set of intervals of X_k .

C,

Let S_k be the smallest set of unit-length closed intervals of X_k , and let $I'_1 = [x'_1, x'_1+1]$ be the first interval in S_k . If $I'_1 = I_1$, then we are done, since we have shown that I_1 is in some smallest set of intervals of X_k .

$$I'_1 = [x'_1, x'_1+1], \text{ and } I_1 = [x_1, x_1+1].$$

If $x'_1 \neq x_1$:

If $x'_1 > x_1$, we can say S_k is not a solution for the question since the first point x_1 isn't included in S_k .

If $x'_1 < x_1$, as x_1 is the leftmost point, there are no points from X_k contained in the interval $[x'_1, x_1]$. Therefore, we could simply replace the interval $[x'_1, x'_1+1]$ in S_k (which is S_{opt}) with the interval $[x_1, x_1+1]$ such that the new set of intervals $S_{k-revise}$ is still optimal (as $|S_k| = |S_{k-revise}|$ and all points are covered).

D,

Recursive:

s = set of intervals (solution)

X = set of the given points

initial $s = \Phi$

Unit-length-closed-intervals_ Recursive (s, X):

1, if $X = \Phi$

2, return s

3,else:

4, sort(X)

5, first_point $\leftarrow X[0]$

6, $I_k = [\text{first_point}, \text{first_point} + 1]$

7, $s \leftarrow (s \cup \{I_k\})$

8, new_ $X \leftarrow (X - \{\text{points: points} \in I_k\})$

9, return Unit-length-closed-intervals($s, \text{new_}X$)

```
1 def Unit_length_closed_intervals(s, X):
2     new_x = list()
3     if len(X) == 0:
4         return s
5     else:
6         X.sort()
7         first_point = X[0]
8         #print(first_point)
9         answer = [first_point, first_point+1]
10        s.append(answer)
11
12        for i in range(0, len(X)):
13            if X[i] <= (first_point+1):
14                continue
15            else:
16                new_x.append(X[i])
17        return Unit_length_closed_intervals(s, new_x)
18
19 X=[6,1,12,3,14,5,5.3,4,5.5,1.3,2.1]
20
21 s = []
22 Unit_length_closed_intervals(s,X)
23 print(s)
```

Run greedy

/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/wushuo/PycharmProjects/cse541/greedy.py
[[1, 2], [2.1, 3.1], [4, 5], [5.3, 6.3], [12, 13], [14, 15]]

Process finished with exit code 0

E,

Iterative:

s = set of intervals (solution)

X = set of the given points

initial s = Φ

Unit-length-closed-intervals_ Iterative(s, X):

```

1,  sort(X)
2,  while X !=  $\Phi$ :
3,      for each point  $p_t$  in X:
4,          first_point <- X[0]
5,           $l_k = [first\_point, first\_point + 1]$ 
6,           $s = (s \cup \{l_k\})$ 
7,           $X = (X - \{points: points \in l_k\})$ 
8,  return s

```

```

def Unit_length_closed_intervals_Iterative(s, X):
    X.sort()
    while len(X) != 0:
        #print(i)
        first_point = X[0]
        new_x = list()
        for i in range(0, len(X)):
            if X[i] <= (first_point + 1):
                continue
            else:
                new_x.append(X[i])
        X = new_x
        answer = [first_point, first_point+1]
        s.append(answer)
    return s

X=[6,1,12,3,14,5,5.3,4,5.5,1.3,2.1]

s = []
s2 = []
Unit_length_closed_intervals(s2,X)
print(s2)
Unit_length_closed_intervals_Iterative(s,X)
print(s)

```

Unit_length_clo... > while len(X) != 0

Run: greedy greedy

/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/wushuo/PycharmProjects/cse541/greedy.py

```

[[1, 2], [2.1, 3.1], [4, 5], [5.3, 6.3], [12, 13], [14, 15]]
[[1, 2], [2.1, 3.1], [4, 5], [5.3, 6.3], [12, 13], [14, 15]]

```

Process finished with exit code 0