Algorithm Design and Analysis

201818018670068

1. Greedy Algorithm

1.1. Natural language AND Pseudo-code.

for every node whose degree is d, if it can become a part of some graph,it must connect with d other nodes with d edges. So we can assume those nodes whose degree is d_1,d_2….,d_n can exist in the same graph.If we delete a node with d degree from this graph, there must exist d nodes whose degree can reduce 1.So we can try delete all of nodes of the graph,if there exist a node that can't fit the rule above, those nodes can't be exist in the same graph.Otherwise there exists an undirected graph $G = (V, E)$ whose node degrees are precisely the numbers d_1,d_2….,d_n.

---

function $IsGraph(ArrayOfDegree)$
1: **for**(every d in D):
2: D->delete() //find **d** nodes in D whose degree is not 0 and make its degree –
3: **while**(d > 0 and i < D->len())
4: i=0
5: **if** (D[i] > 0) **then** :
6: D[i]–
7: d–
8: **end**
9: i++
10: **end**
11: **if**(d > 0) **then** : //if we can't find **d** nodes, it can't be a graph
12: **return** false
13: **end**
14:**end**
15:**return** true // we find for every d in D,so it can become a graph

---

1.2. Describe the greedy-choice property and optimal substructure.

The greedy-choice property is if the remaining nodes can compose a graph when a node is deleted from the graph.

Optimal substructure:

$$\begin{Bmatrix} OPT(d - d[0]) \\ 0 \end{Bmatrix}$$

1.3. Prove the correctness of your algorithm.

for every node whose degree is d, if it can become a part of some graph,it must connect with d other nodes with d edges. So we can assume those nodes whose degree is d_1,d_2....,d_n can exist in the same graph. If we delete a node with d degree from this graph, there must exist d nodes whose degree can reduce 1. So we can try delete all of nodes of the graph,if there exist a node that can't fit the rule above, those nodes can't be exist in the same graph.Otherwise there exists an undirected graph $G = (V, E)$ whose node degrees are precisely the numbers d_1,d_2....,d_n.

1.4. Analyse the complexity of your algorithm.

Time Complexity is $O(nd)$ where d is the average value.

2. Greedy Algorithm

2.1. Natural language AND Pseudo-code.

Compare the first char of string s and t,if the two chars are different,delete the first char from s. Otherwise delete the first char from t. Repeat it until s or t don't contain any char. In the end, if the length of t is 0,t is a subsequence of s, otherwise,t is not a subsequence of s.

---

function $SubsequenceOf(s, t)$
1: **while**(s->length != 0 or t->length != 0)
2: **if** (s->first == t->first) **then**:
3: s->delete(s->first)
4: t->delete(s->first)
5: **else**
6: s->delete(s->first)
7: **end**
8: **end**
9: **if** (t->length == 0) **then** :
10: **return** true

11: **else** :
12: **return** false

2.2. Describe the greedy-choice property and optimal substructure.

The greedy-choice is if the first characters of s and t are same.

Optimal substructure:

$$\begin{Bmatrix} OPT(s - s[0], t) \\ OPT(s - s[0], t - t[0]) \end{Bmatrix}$$

2.3. Prove the correctness of your algorithm.

Compare the first char of string s and t,if the two chars are different,delete the first char from s. Otherwise delete the first char from t. Repeat it until s or t don't contain any char. In the end, if the length of t is 0,t is a subsequence of s, otherwise,t is not a subsequence of s.

2.4. Analyse the complexity of your algorithm.

Time Complexity is $O(LengthOfS)$

3. Greedy Algorithm

3.1. Natural language AND Pseudo-code.

We need compute $\lfloor \frac{n}{m} \rfloor$ or $\lceil \frac{n}{m} \rceil$ as the length of the first part of the nope we cut. And we need update $n = n$ - $\lfloor \frac{n}{m} \rfloor$ or $\lceil \frac{n}{m} \rceil$ and $m = m$ - 1, the new rope is the rope without the first part we cut from the rope. So we can repeate previous steps until the length of the rope is 0. And We get the maximum product.

---

function *maximum_product*(m, n)
1: **while** (n is not 0 **or** m == 0)
2: product = n/m * product
3: n = n - n/m
4: m = m - 1
5: **end**

3.2. Describe the greedy-choice property and optimal substructure.

The greedy-choice is cutting $\lfloor \frac{n}{m} \rfloor$ or $\lceil \frac{n}{m} \rceil$ everytime.

Optimal substructure: $OPT(n - \lfloor \frac{n}{m} \rfloor, m - 1) * \lfloor \frac{n}{m} \rfloor$

3.3. Prove the correctness of your algorithm.

The maximum product must be the product of the value closest to the average so we compute
$\lfloor \frac{n}{m} \rfloor$ or $\lceil \frac{n}{m} \rceil$ everytime.

3.4. Analyse the complexity of your algorithm.

Time Complexity is $O(m)$