

学号： 201618013229011
姓名： 李坚松

Assignment 3

Problem-1

Solution

We suppose that the graph does exist, where $d_1 \geq d_2 \geq d_3 \geq \dots \geq d_n$, we mark it as $G\{d_1, d_2, d_3, \dots, d_n\}$, then we can construct an equivalent graph $G'\{d_2-1, d_3-1, \dots, d_{d_1+1}-1, d_{d_1+2}, d_{d_1+3}, \dots, d_n\}$, based on this greedy rules, we can get series of sub-problems. For each sub-problem, if any $d_i < 0$ occurs, it means that the graph doesn't exist. However, if all the elements are 0 in one subsequence eventually, it means that the graph does exist. Following is pseudo code of the algorithm:

Algorithm: GraphJudge (int A[], int n)

Input: An array A that consists of n positive integers

Output: whether the n integers can construct an undirected graph

Begin

```
sort(A); // sort the degree array in decrease order
if( A[0] = 0 )
    return true;
else
{
    if( A[n-1] < 0 )
        return false;
    else
    {
        int d = A[0]; // maximum of degree
        n = n - 1;
        A = A - A[0]; // sub-problem
        for( int j = 0; j < d; ++j )
            A[j] = A[j] - 1;
        GraphJudge( A, n );
    }
}
```

End

Proof

We should prove the fact that if $d_1 \geq d_2 \geq d_3 \geq \dots \geq d_n$, we mark the graph as $G\{d_1, d_2, d_3, \dots, d_n\}$, then we can construct an equivalent graph

$G'\{d_2-1, d_3-1, \dots, d_{d_1+1}-1, d_{d_1+2}, d_{d_1+3}, \dots, d_n\}$. To prove that $G' \rightarrow G$, we can use the following construction rules, add a vertex and add edges from this vertex to the previous d_1 vertices, then we can get a graph that has the same required degrees with G . Similarly we can prove that $G \rightarrow G'$. Based on the fact above, we can know that in one step we can get a subsequence, if the smallest degree is less than 0, it means that the graph doesn't exist at all, else the degrees should be 0 eventually.

Time Complexity

For each sort operation, it takes $O(n \log n)$ time, at the worst case, it will take $O(n)$ iteration, so the time complexity is $O(n^2 \log n)$.

Problem-4

Solution

The problem is to let us to maximize the payoff. If the two sets are in increasing order, by this greedy rule, we can get maximum of the payoff. The pseudo code of the algorithm is shown below:

Algorithm: GetMaxPayoff (int A[], int B[])

Input: two sets marked by A and B**Output:** reorder the two sets to get the max payoff

Begin

Sort A by increasing order;

Sort B by increasing order;

End

Proof

By this greedy rule, we can know that if $a_i < a_j$, then $b_i < b_j$, where $i \neq j$. Suppose that there exist an $i \neq j$ that $a_i < a_j$, and $b_i > b_j$, then

$$a_i^{b_j} a_j^{b_i} = a_i^{b_j} (a_j^{b_i - b_j} a_j^{b_j}) < (a_i^{b_j} a_i^{b_i - b_j}) a_j^{b_j} = a_i^{b_i} a_j^{b_j}$$

So i and j doesn't exist, the greedy rule is correct.

Time Complexity

It takes two sort operations, therefore the time complexity is $O(n \log n)$.










Problem-5

Implement

Implement see Huffman.cpp.

Result

The input files characters are all in the form of ASCII, the output compression are in the form of binary, however to make them visible, we output them in ASCII, so the size of compression file should be one in eight of the output files' size. The compression result is shown below:

 Aesop_Fables.txt	2016/10/17 12:12	文本文档	186 KB
 Aesop_Fables-de.txt	2016/10/17 12:12	文本文档	186 KB
 Aesop_Fables-en.txt	2016/10/26 22:15	文本文档	823 KB
 gmon.out	2016/10/26 22:24	OUT 文件	85 KB
 graph.txt	2016/10/17 12:13	文本文档	2,198 KB
 graph-de.txt	2016/10/17 12:13	文本文档	2,198 KB
 graph-en.txt	2016/10/26 22:24	文本文档	7,269 KB
 Huffman.cpp	2016/10/26 22:14	C++ Source File	7 KB
 Huffman.exe	2016/10/26 22:14	应用程序	1,960 KB

So for the file Aesop_Fable.txt, the compression ratio is $823/(8*186)=55.3\%$, while for the file graph.txt, the compression ratio is $7269/(8*2198)=41.3\%$.