演算法 PA2

B10901176 蔡弘祥

EDA union lab machines port:40062

1. Data Structure

   The input file is composed of several lines. The first line is a number which tell us the number of points on the circle, let's call it N. After that, N/2 lines will come into our sight, telling us what are the two endpoints of each line. We declare a array called "data" with length N that save the point which it connects to. For example, if 0,2 form a line, we save data[0]=2 and data[2]=0. Next, we declare "M[N][N]" and "S[N][N]". M[i][j] is used to record the recursive function "mps" so that we can save a great amount of time recalculating the same thing in the recursion. On the other hand, S[i][j] is used to record which case we have chosen to do the recursion (There are four cases in my recursion. To get more info, see the next paragraph.). Finally, the array "result" is declared to save those point we need to select. Meanwhile, the array "sortResult" is basically same as the "result", but sorted in an increasing order (better output).

2. Method

   i. Original Method

      The original method is quite intuitive.

      The recursive function is written as :

      M(i,j) = max{[max(M(i,k) + M(k+1,j)) for i<=k<j] , M(i+1,j-1) + 1 }

      The base case is :

      M(i,j) = 0 for i == j,

      M(i,j) = 0 for i+1 == j , if i is not connected to j,

      M(i,j) = 1 for i+1 == j , if i is connected to j.

      However, the running time is O(n^3), which is not acceptable when the input is large. Thus, the New Method appears.

   ii. New Method

      The recursive function is written as (case x is written in S[i][j] used to trace back) :

      M(i,j) = 0 if i>=j,

      M(i,j) = M(i+1,j-1) ,

      if i is connected to j (case 0),

      M(i,j) = M(i,j-1) ,

      if the point that connects to j is not in the region of I to j (case 1),

      M(i,j) = max{M(i,j-1), M(i,data[j]-1) + M(data[j]+1,j-1) + 1},

      if the point that connects to j is in the region of I to j.

      (case 2 if choosing the former one, case 3 if choosing the latter one)

3. Other Findings

   One important finding is that dynamic array performs better than stl:vector due to the locality of the program. My first try is implementing the idea using vector in every array, but I found that it doesn't perform well even in the case of 1000.in. After asking someone who knows more about this sort of things, I understand that dynamic array uses a consecutive memory while the vector does not. Consecutive memory usage can significantly increase the efficiency of the whole program. Thus I rewrite the part which used vector and replace it with dynamic array.