Consider the following problem. Given an array A consisting of n distinct integers A[1], ... A[n]. It is known that there is a position p , such that A[1], .., A[p] is in increasing order and A[p], A[p+1], ..., A[n] is in decreasing order

1、Write a brute force algorithm to find the position p. What is the time complexity of

your algorithm

For i=0 to n

Do: if (A[i]>A[i+1]&&A[i]<A[i-1])

Then report “P is “ + i;

End

Time complex is n

1. Devise a “divide and conquer" algorithm to find the position p.

divide and conquer (A[];begin; last)

Begin:

Int mid = (1+n)/2;

If (A[mid]>A[mid+1]&&A[mid]<A[mid-1])

Return true;

Else if (A[mid]>A[mid+1]&&A[mid]>=A[mid-1])

return divide and conquer（ A, first, mid-1;）

Else if(A[mid]<=A[mid+1 &&A[mid]<A[mid-1])

return divide and conquer(A, mid-1, last)

end

3、Set up a recurrence relation for the number of comparisons made by your algorithm

and explain it

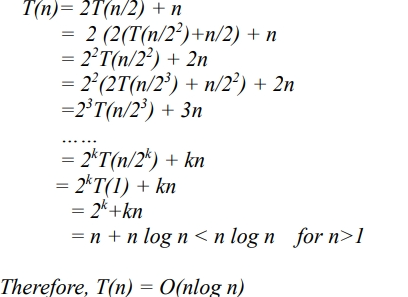
Tn=1(n=0)

Tn=T(n/2)+1(n>0)

We first divide the array into half and compare the requirement and our middle number, if do not fit, we will repeat this process and chose which half to be return, until we find the proper answer or there is only one element. Since in the end we still need to compare so we need to plus O(1) in the end.

1. Based on the recurrence relation, show the complexity of your algorithm in big-O notation and prove it using either the iterative method or the substitution method, i.e., Mathematical Induction (for simplicity, you can assume that n = 2k).

这里就直接用数学归纳法一直代换就好，具体过程较长，但是是ttl有说过的。



参考这个，不过我们需要将第一行中的n代换为1.

**Q2:1：Briefly describe the idea of the polynomial time reduction. Explain how to use it to**

**prove a problem is NP complete.**

Reduction means a problem can be transport into a larger problem in polynomial time. In the

establish of NPC problem, we usually first find a problem, and try to do reduction to this

problem to the problem we want to prove. Since a npc problem means any np problem can be

reduct into it, if we prove the problem we find is np, then the origional problem is npc**.**

**2：4-SAT Problem: for a Boolean formula in CNF in which each clause has exactly 4 literals, determine if there is an assignment of Boolean value to its variables so that the formula evaluates to true? (i.e., the formula is satisfiable). Prove 4-SAT Problem is NP-Complete**

**这个题两个思路，一个是把它按照我pdf里面这一章说的转化成图进行运算，第二种就直接简单粗暴的列举一下，直接说明3-SAT可以归约到这里，因为3-SAT是我们上课直接讲过的NP问题，我们可以直接说明4-SAT是NPC问题。**