2020/05/02--2020/05/08

团队训练

暂无

王瑞琦

专题

1. 树

还没有打比赛orz□稍微练了几道树的题

冯宇扬

专题

1. lct复健

比赛

1. 没打

常程

专题

- 尺取法
- 前缀和

（由于我太菜了所以写一点基础的）

比赛

暂时没打

本周推荐

王瑞琦

给出一个以=结尾的包含+-*/%()的式子，按运算规则计算其值。

给出的式子是中缀表达式，将其转化为后缀表达式求解。

转换方法：

用队列存储运算符，栈辅助处理。对每个字符，规则如下：

1）若为数字，直接入队（当然，多位数字要提前处理一下）

2）若为运算符：

1.栈为空直接入栈

2.优先级大于栈顶元素入栈

3.优先级小于等于栈顶元素，则栈顶元素出栈入队，直到优先级大于栈顶元素。

3）若为（，直接入栈

4）若为），栈顶元素出栈入队，直到（，两个括号不入队。

计算方法：

对于一个后缀表达式，用栈辅助计算

从左至右扫描，若为数字，直接入栈，若为运算符，则将栈顶两个元素计算，计算结果入栈。

直到最后留下一个数字，即为结果。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char stack[500];
int queue[500];
void calcu(int r,int top)
{
      switch (stack[top]){
      case '+':queue[r-1]=queue[r]+queue[r-1];break;
      case '-':queue[r-1]=queue[r-1]-queue[r];break;
      case '*':queue[r-1]=queue[r]*queue[r-1];break;
      case '/':queue[r-1]=queue[r-1]/queue[r];break;
      case '%':queue[r-1]=queue[r-1]%queue[r];break;
      }
}
int main()
{
        char s[500];
```

```
  int pr[300]={0};
  pr[(int)'+']=pr[(int)'-']=1;
  pr[(int)'*']=pr[(int)'/']=pr[(int)'%']=2;
  pr[(int)'(']=0;
  gets(s);
  int top=0,r=0,i=0;
  while (s[i]!='=')
  {
      if (s[i]>='0'&&s[i]<='9')
      {
          int x=(int)s[i]-48;
          i++;
          while (s[i]>='0'&&s[i]<='9')
          {
              x=x*10+(int)s[i]-48;
              i++;
          }
          queue[++r]=x;
          i--;
      }
      else if (s[i]!='('&&s[i]!=')'&&s[i]!=' '){
          if (top==0) stack[++top]=s[i];
          else {
              if (pr[(int)s[i]]>pr[(int)stack[top]]) stack[++top]=s[i];
              else {
                  while (top!=0&&pr[(int)s[i]]<=pr[(int)stack[top]])
calcu(r--,top--);
                  stack[++top]=s[i];
              }
          }
      }
      else if (s[i]=='(') stack[++top]=s[i];
      else if (s[i]==')') {
              while (stack[top]!='(') calcu(r--,top--);
              top--;
      }
      i++;
  }
  for (int i=top;i>=1;i--) calcu(r--,top--);
  printf("%d",queue[1]);
  return 0;
}
```

冯宇扬

搓了一道 P1600 天天爱跑步

发现了绝妙的做法，但是还没有debug出来

思路是求完dfn以后转化成三维偏序，然后能压掉一维

常程

poj 2566 （专题里有，不过我觉得挺好的）

给定一个数组和一个值t，求一个子区间使得其和的绝对值与t的差值最小。数组中的数可正可负。

将前缀和排序，排序后每两个位置的前缀和之差代表一个原来的区间的和的绝对值(妙啊），同时在这个排序后的前缀和序列中随区间增缩所取的区间和的绝对值也增缩。这样运用尺取法是一个好的选择。

```
#include<cstdio>
#include<algorithm>
#define INF 0x3f3f3f3f
using namespace std;
int n,k,a;
typedef pair<int,int> unit;
unit sum[100005];
int main()
{
    while(scanf("%d%d",&n,&k)&&n)
    {
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&a);
            sum[i]=unit(sum[i-1].first+a,i);
        }
        sort(sum+1,sum+1+n);
        for(int i=1;i<=k;i++)
        {
            scanf("%d",&a);
            int t=INF,l,r;
            for(int left=1,right=1;right<=n;right++)
            {
                while(sum[right].first-sum[left].first>a)
                {
                    if(sum[right].first-sum[left].first-a<t)
                    {
                        t=sum[right].first-sum[left].first-a;
                        l=sum[left].second;
                        r=sum[right].second;
                    }
                    left++;
                }
                if(a-sum[right].first+sum[left].first<t)
                {
                    t=a-sum[right].first+sum[left].first;
                    l=sum[left].second;
                    r=sum[right].second;
                }
            }
```

```
                printf("%d %d %d\n",sum[r].first-sum[l].first,l,r);
            }
        }
        return 0;
    }
```