这题我们可以用动态规划 f[s,0],f[s,1]分别表示表达式值为0和1的方案数。

容易得出：

f[a+b,0]=f[a,0]\*f[b,0] (乘法规律）

f[a+b,1]=f[a,0]\*f[b,1]+f[a,1]\*f[b,0]+f[a,1]\*f[b,1]

f[a\*b,0]=f[a,0]\*f[b,0]+f[a,0]\*f[b,1]+f[a,1]\*f[b,0]

f[a\*b,1]=f[a,1]\*f[b,1];

使用栈模拟

判断空位

右括号')'后无空位，其他运算符后都有一个空位

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63 | #include <iostream>  #include <cstdio>  #include <cstdlib>  #include <cstring>  #include <cmath>  #include <iomanip>  #include <algorithm>  const int inf=0x7fffffff;  using namespace std;  struct dps  {  int a[2];//保存0或1的方案数  } dp[100005];  char opt[100005], str[100003];  inline void calc(char op, dps &a, dps &b)  {  if (op == '+')  {  a.a[1] = (a.a[1] \* (b.a[0] + b.a[1]) + a.a[0] \* b.a[1]) % 10007;  a.a[0] = a.a[0] \* b.a[0] % 10007;  }  else  {  a.a[0] = (a.a[0] \* (b.a[0] + b.a[1]) + a.a[1] \* b.a[0]) % 10007;  a.a[1] = a.a[1] \* b.a[1] % 10007;  }  }  int main()  {  freopen("exp.in" ,"r", stdin);  freopen("exp.out", "w", stdout);  int len, Opti = 1, Fsi = 1;  cin>>len>>str;  opt[1] = '('; //为方便，先给一个左括号  str[len] = ')'; //为方便，末尾先给一个右括号  dp[1] = dps {1,1};//(１，１)表示０的方案一个，１的方案一个，因为都有一种可能  for (int i=0; i<=len; i++)//从左到右扫描表达式  if (str[i] == '(')//如果为左括号  opt[++Opti] = '('; //相应表达式也为左括号  else if (str[i] == ')') //如果为右括号  {  for (; opt[Opti] != '('; --Opti, --Fsi)//一直往回找到相匹配的左括号  calc(opt[Opti], dp[Fsi - 1], dp[Fsi]);//参数为操作符，结构体前一个值和后一个值  --Opti;//弹出操作符  }  else//不是左括号也不是右括号，那就是运算符  {  //括号内找到优先级别高的运算符进行运算  for (; (opt[Opti] <= str[i]) && (opt[Opti] != '('); --Opti, --Fsi)//\*:42,+:43  calc(opt[Opti], dp[Fsi - 1], dp[Fsi]);//动规  opt[++Opti] = str[i];  dp[++Fsi] = dps {1,1};  }  cout<<dp[1].a[0]<<endl;  return 0;  } |