

Wintercoder

Wintercoder

第一阶段

第二阶段

矩阵快速幂3375

KMP3375

Trie2580

计算器的改良:1022

栈(后缀表达式):1449

单调队列:1886

并查集:3367

康托展开:5367

最长不下降子链 :3902

背包问题:1616

NIM博弈:2197

第一阶段

string.h 运行截图。

```
1  #include "String.h"
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6      String s1("hello "); //构造函数
7      String s2("world"); //构造函数
8      String s3(s2); //拷贝函数
9      cout << s3 << endl; //输出符号重载 s3 = s1;
10     cout << s3 << endl;
11     cout << s1 + s2 << endl; //实现"+"的重载 效果:
12 }
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
(drqa) cz@czdeMacBook-Pro:~/wintercoder/section1$ ./String.out
world
world
hello world
```

第二阶段

矩阵快速幂3375

思路简述

由于矩阵乘法满足结合律，对矩阵乘法操作进行重载后，进行快速幂运算就可以啦。

快速幂大概就是利用之前的计算结果，幂次从小到大，比如在计算 2^8 时，直接用 $2^4 \times 2^4$ 就可以而不是2乘8次：

$$\mathbf{A}^k = \mathbf{A}^{2^0} \times \mathbf{A}^{2^1} \times \mathbf{A}^{2^2} \times \dots \times \mathbf{A}^{2^n}$$

截图

R45678341 记录详情

编程语言
C++

代码长度
2.06KB

用时
260ms

内存
4.12MB

测试点信息

源代码

测试点信息

#1
AC
29ms/2.44MB

#2
AC
61ms/3.98MB

#3
AC
31ms/2.63MB

#4
AC
28ms/2.49MB

#5
AC
29ms/2.62MB

#6
AC
63ms/4.12MB

#7
AC
2ms/512.00KB

#8
AC
2ms/772.00KB

#9
AC
9ms/1.38MB

#10
AC
6ms/1.22MB

所属题目
P3390 【模板】矩阵快速幂

评测状态
Accepted

评测分数
100

提交时间
2021-01-28 16:56:03

代码

```
#include<iostream>
#include<fstream>
#define ll long long
#define mo 1000000007
using namespace std;

class Matrix{
public:
    Matrix(ll n){
        length = n;
        this->a = new ll*[n];
        for(ll i=0;i<n;i++)
        {
            this->a[i] = new ll[n];
        }
        for(ll i=0;i<n;i++)
        {
            for(ll j=0;j<n;j++)
            {
                this->a[i][j] = 0;
            }
        }
    }
    void identityMatrix()
    {
```

```

        for(ll i=0;i<length;i++)
        {
            a[i][i] = 1;
        }
    }
    Matrix operator*(const Matrix &b)
    {
        Matrix c = Matrix(length);
        for(ll i=0;i<length;i++)
        {
            for(ll j=0;j<length;j++)
            {
                for(ll k = 0;k<length;k++)
                {
                    c.a[i][j] += (this->a[i][k] * b.a[k][j]) % mo;
                }
                c.a[i][j] = c.a[i][j] % mo ;
            }
        }
        return c;
    }
    ll length;
    ll **a ;
};

class MatrixMultiSolution
{
private:
    ll n;
    ll k;
public:
    MatrixMultiSolution()
    {
        Matrix A = inputFunc();
        Matrix res = mySolution(A);
        outputFunc(res);
    }
    Matrix mySolution(Matrix A)
    {
        Matrix res(A.length);
        res.identityMatrix();
        while(k)
        {
            if(k & 1)
                res = res * A;
            A = A*A;
            k>>=1;
        }
        return res;
    }
}

```

```

Matrix inputFunc()
{
    cin>>this->n>>this->k;
    Matrix A(n);
    for(ll i=0;i<n;i++)
    {
        for(ll j=0;j<n;j++)
        {
            cin>>A.a[i][j];
        }
    }
    return A;
}

void outputFunc(Matrix A)
{
    for(ll i=0;i<this->n;i++)
    {
        for(ll j=0;j<this->n;j++)
        {
            cout<<A.a[i][j]<<" ";
        }
        cout<<endl;
    }
}

};

int main()
{
    // ifstream fin("in.txt");
    // ofstream fout("out.txt");
    // cin.rdbuf(fin.rdbuf());
    // cout.rdbuf(fout.rdbuf());
    MatrixMultiSolution s ;
    return 0;
}

```

KMP3375

思路简述

字符串匹配的快速方法，时间复杂度从暴力的 $O(n*m)$ 降为 $O(n+m)$ 。思路是，对于模式串而言，构造一个KMP数组，在于目标串对比时失配时，可以根据KMP数组将模式串的位置跳转到 $KMP[j]$ 。可以这样操作的依据是KMP数组记录了最大真前缀和真后缀的信息，即如果 $KMP[i] = j$ 表示模式串 $1\sim j$ 的前缀和 $i-j+1\sim i$ 的后缀是相同的。

举个例子：

pattern: abcabc

target: abcabdababcabc

失配后可以移动为：

abcbabc

abcbabdababcbabc

通过计算一个KMP数组在失配时进行跳转，可以大大减少匹配的复杂度。而KMP数值可以通过模式串和自身的匹配得到。

截图

测试点信息

源代码

测试点信息

Subtask #0


#1 AC 4ms/472.00KB	#2 AC 5ms/620.00KB	#3 AC 6ms/592.00KB
--------------------------	--------------------------	--------------------------

Subtask #1

#4 AC 6ms/620.00KB	#5 AC 4ms/484.00KB	#6 AC 4ms/620.00KB	#7 AC 4ms/620.00KB
--------------------------	--------------------------	--------------------------	--------------------------

Subtask #2

#8 AC 20ms/2.59MB	#9 AC 19ms/2.61MB	#10 AC 19ms/2.58MB	#11 AC 593ms/2.59MB	#12 AC 21ms/2.55MB	#13 AC 571ms/2.71MB
-------------------------	-------------------------	--------------------------	---------------------------	--------------------------	---------------------------

 cz_nju

所属题目 P3375 【模板】KMP字符串匹配

评测状态 Accepted

评测分数 100

提交时间 2021-02-11 10:51:42

代码

```
#include<iostream>
#include<cstring>
using namespace std;
class KMPSolution
{
public:
    KMPSolution()
    {
        string target;
        string pattern;
        cin>>target;
        cin>>pattern;
        int lt = target.length();
        int lp = pattern.length();
        target = "0"+target;
        pattern = "0"+ pattern;
        int *kmp = new int[lp+1];
        memset(kmp,0,sizeof(int)*(lp+1));
        int j=0;
        for(int i=2;i<=lp;i++)
        {
            while(j && (pattern[j+1] != pattern[i]))
                j = kmp[j];
            if(pattern[j+1] == pattern[i])
                j++;
        }
    }
};
```

```

        kmp[i] = j;
    }
    j = 0;
    for(int i=1;i<=lt;i++)
    {
        while(j && pattern[j+1] != target[i])
        {
            j = kmp[j];
        }
        if(pattern[j+1] == target[i])
            j++;
        if(j == lp)
        {
            cout<<i-lp+1<<endl;
            j=kmp[j];
        }
    }
    for (int i=1;i<=lp;i++)
        cout<<kmp[i]<<" ";
}

};
int main()
{
    KMPSolution x;
    return 0;
}

```

Trie2580

思路简述

截图

R45816442 记录详情

编程语言	代码长度	用时	内存
C++	1.98KB	829ms	58.46MB

[测试点信息](#) [源代码](#)

测试点信息

#1 AC 3ms/484.00KB	#2 AC 198ms/58.29MB	#3 AC 3ms/624.00KB	#4 AC 3ms/600.00KB	#5 AC 24ms/4.36MB	#6 AC 45ms/58.16MB
#7 AC 77ms/57.15MB	#8 AC 82ms/56.80MB	#9 AC 236ms/58.42MB	#10 AC 158ms/58.46MB		



所属题目 [P2580 于是他错误的点名开始了](#)

评测状态 Accepted

评测分数 100

提交时间 2021-01-30 21:11:23

代码

```
#include<iostream>
#include<string>
#include<vector>
#include<string.h>
#include<fstream>
using namespace std;
struct TrieNode
{
    int cnt;
    int id;
    int child[26];
    TrieNode(int id)
    {
        this->id = id;
        cnt = 0;
        memset(child, 0, sizeof(child));
    }
};
class Trie
{
    vector<TrieNode> nodeList;

public:
    Trie()
    {
        TrieNode rootNode = TrieNode(0);
        nodeList.push_back(rootNode);
    }
    void insert(string name)
    {
        int headId = 0;
        for (int i = 0; i<name.size(); i++)
        {
            char curCh = name[i];
            int chId = curCh - 'a';
            if (nodeList[headId].child[chId] == 0)
            {
                int nodeId = nodeList.size();
                TrieNode newNode = TrieNode(nodeId);
                nodeList.push_back(newNode);
                nodeList[headId].child[chId] = nodeId;
                headId = nodeId;
            }
            else
```

```

        {
            headId = nodeList[headId].child[chId];
        }
    }
}

int search(string name)
{
    int headId = 0;
    for (int i = 0; i < name.size(); i++)
    {
        int chId = name[i] - 'a';
        if (nodeList[headId].child[chId] == 0)
            return 0;
        else
        {
            headId = nodeList[headId].child[chId];
        }
    }
    nodeList[headId].cnt += 1;
    if (nodeList[headId].cnt == 1)
    {
        return 1;
    }
    else
        return 2;
}
};

class Solution
{
public:
    void TrieSolution()
    {
        Trie trie = Trie();
        int n;
        cin >> n;
        for (int i = 0; i < n; i++)
        {
            string name;
            cin >> name;
            trie.insert(name);
        }
        int m;
        cin >> m;
        for (int i = 0; i < m; i++)
        {
            string name;
            cin >> name;

            int result = trie.search(name);

```



```

        if (result == 0)
        {
            cout << "WRONG\n";
        }
        else if (result == 1)
        {
            cout << "OK\n";
        }
        else
        {
            cout << "REPEAT\n";
        }
    }
}
};

int main()
{
    // ifstream fin("in.txt");
    // ofstream fout("out.txt");
    // cin.rdbuf(fin.rdbuf());
    //cout.rdbuf(fout.rdbuf());
    Solution s = Solution();
    s.TrieSolution();
    //不要用指针操作vector中的结构体，是真的迷。比如下面的操作
    // vector<TrieNode> nodeList;
    // TrieNode p = TrieNode(0);
    // nodeList.push_back(p);
    // TrieNode *head = &nodeList[0];
    // head->child[8] = 555;
    // cout<<nodeList[0].child[8];
}

```

计算器的改良:1022

思路简述

思路比较简单，就是细节比较繁琐，我这里尝试使用了正则表达式

```

string num_pattern = "\\-*\\+*[0-9\\.]+(?! [a-zA-Z0-9])";
string var_pattern = "\\-*\\+*[0-9\\.]*[a-zA-Z]";

```

分别来识别常数和变量，然后分别处理变量前的系数和常数的系数，讲其存入向量中。计算得出答案。其中一些边界情况如 -a , a 这样的需要分情况处理得到系数。

截图

R46227510 记录详情

编程语言	代码长度	用时	内存
C++11	2.40KB	18ms	636.00KB

[测试点信息](#) [源代码](#)

测试点信息

#1	#2	#3	#4	#5	#6
AC	AC	AC	AC	AC	AC
3ms/624.00KB	3ms/496.00KB	3ms/628.00KB	3ms/632.00KB	3ms/636.00KB	3ms/628.00KB

所属题目 [P1022 \[NOIP2000 普及组\] 计算器的改良](#)评测状态 Accepted评测分数 100

提交时间 2021-02-05 20:13:48

代码

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
#include<fstream>
#include <typeinfo>
#include <sstream>
#include <regex>

using namespace std;

class Solution
{
public:
    string identifyFunc(string s,vector<int > & num,vector<int > & var)
    {
        string num_pattern = "\\-*\\++[0-9\\.]+(?![a-zA-Z0-9])";
        string var_pattern = "\\-*\\++[0-9\\.]*[a-zA-Z]";
        regex num_p(num_pattern);
        regex var_p(var_pattern);
        smatch m;
        string s1 = s;
        while (regex_search(s, m, num_p))
        {
            for (auto x : m)
            {
                int cur = atoi(x.str().c_str());
                //cout << "num:" << cur << " ";
                num.push_back(cur);
            }
            s = m.suffix().str();
        }
        string var_name = "";
        while (regex_search(s1, m, var_p))
        {
            for (auto x : m)
```

```

{
    int cur;
    if (x.str().length() > 2)
        cur = atoi(x.str().c_str());
    else if (x.str()[0] == '-')
        cur = -1;
    else if (x.str()[0] == '+')
        cur = 1;
    else if (x.str().length() == 1)
        cur = 1;
    else
        cur = atoi(x.str().c_str());
    var_name = x.str()[x.str().length() - 1];
    //cout << "var:" << cur << " ";
    var.push_back(cur);
}
s1 = m.suffix().str();
}
return var_name;
}
Solution(string input)
{
    string split = "=";
    int split_postion = input.find(split, 0);
    //cout << split_postion << endl;
    string left = input.substr(0, split_postion);
    string right = input.substr(split_postion + 1);
    //cout << left << " = " << right << endl;
    vector<int> left_num;
    vector<int> left_var;
    vector<int> right_num;
    vector<int> right_var;
    string var_name = "";
    string var_n1 = identifyFunc(left, left_num, left_var);
    string var_n2 = identifyFunc(right, right_num, right_var);
    int var = 0;
    for (auto v : left_var)
        var += v;
    for (auto v : right_var)
        var -= v;
    int num = 0;
    for (auto v : left_num)
        num -= v;
    for (auto v : right_num)
        num += v;
    //cout << num << " " << var << endl;
    if (var_n1.length() != 0)
        var_name = var_n1;
    else

```

```

        var_name = var_n2;
        double res = double(num) / double(var);
        cout << var_name << "=";
        printf("%.3f\n", res == 0 ? abs(res) : res);
    }
};

int main ()
{
    //std::string s ("44-3-48x=9889x+8");
    //ifstream fin("P1022_1.in");
    //cin.rdbuf(fin.rdbuf());
    string input;
    cin >> input;
    Solution x(input);
    return 0;
}
//int main()
//{
//    //ifstream fin("in.txt");
//    //ofstream fout("out.txt");
//    //cin.rdbuf(fin.rdbuf());
//    //cout.rdbuf(fout.rdbuf());
//    Solution s = Solution();
//}

```

栈(后缀表达式):1449

思路简述

后缀表达式的标准求解方法。遇到操作数压栈，遇到操作符从栈中pop两个元素运算后压栈。输入全部结束后，栈中元素为最终答案。

截图

R45863454 记录详情

编程语言

C++11

代码长度

2.07KB

用时

16ms

内存

644.00KB

测试点信息

源代码

测试点信息

#1	#2	#3	#4	#5	#6
AC	AC	AC	AC	AC	AC
4ms/616.00KB	2ms/624.00KB	2ms/620.00KB	4ms/632.00KB	2ms/644.00KB	2ms/632.00KB

所属题目

P1449 后缀表达式

评测状态

Accepted

评测分数

100

提交时间

2021-01-31 17:15:36

代码

```

#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
#include<fstream>
using namespace std;

struct unit
{
    int val;
    char opera;
    bool isVal;
    unit(int v)
    {
        this->val = v;
        isVal = true;
    }
    unit(char op)
    {
        opera = op;
        this->isVal = false;
    }
    friend ostream &operator<<(ostream &output,const unit &u)
    {
        output << u.isVal << " " << u.val << " " << u.opera << endl;
        return output;
    }
};

class Solution
{
public:
    vector<unit> inputFunc()
    {
        string str;
        char ops[4] = { '+', '-', '*', '/' };
        vector<char> operators(ops,ops+4);
        cin >> str;
        //cout << str << endl;
        vector<unit> inputs;
        string curNum = "";
        for (int i = 0; i < str.size(); i++)
        {
            if (find(operators.begin(), operators.end(), str[i]) == operators.end())
            {
                //not found
                if (str[i] != '.' && str[i] != '@')
                {
                    curNum += str[i];
                }
            }
        }
    }
};

```

```

        else
        {
            if (curNum != "")
            {
                int num = atoi(curNum.c_str());
                inputs.push_back(unit(num));
                curNum = "";
            }

        }
    }
    else
    {
        inputs.push_back(unit(str[i]));
    }
}

//for (auto t : inputs)
//{
//    cout << t.isVal << " " << t.val<<" " << t.opera << endl;
//    cout << "*****" << endl;
//}
return inputs;
}

void postfixExpression()
{
    vector<unit> inputs = inputFunc();
    vector<unit> resStack;
    for (auto u : inputs)
    {
        if (u.isVal)
            resStack.push_back(u);
        else
        {
            unit op2 = resStack.back();
            resStack.pop_back();
            unit op1 = resStack.back();
            resStack.pop_back();
            unit temp(0);
            if (u.opera == '+')
                temp = unit(op1.val + op2.val);
            else if (u.opera == '-')
                temp = unit(op1.val - op2.val);
            else if (u.opera == '*')
                temp = unit(op1.val * op2.val);
            else if (u.opera == '/')
                temp = unit(op1.val / op2.val);
            //else
            //    cout << "error" << u.opera << endl;
            resStack.push_back(temp);
        }
    }
}

```

```

    }

    }
    cout << resStack.back().val << endl;
};

};

int main()
{
    /*ifstream fin("in.txt");
    ofstream fout("out.txt");
    cin.rdbuf(fin.rdbuf());*/
    //cout.rdbuf(fout.rdbuf());
    Solution s = Solution();
    s.postFixExpression();
}

```

单调队列:1886

思路简述

对于每个窗口最大值维护一个单调递减队列，对于最小值维护一个单调递增队列。以最小值为例。例如窗口大小为3，输入序列为：

1, 2, 3, 4, -1, -2, 3, 4, -7, 8

那么在遍历输入序列的过程中，单调递增队列里面存的始终是之后有可能成为最小值的元素，所以当下一个输入数字大于队列尾部的值时，将当前元素加入单调队列尾部。如果当前元素小于等于单调队列尾部，要对单调队列尾部pop直到队列为空或者队列尾部的值严格小于当前元素。再将当前元素压入队尾。

同时队列头部的元素位置不能小于窗口启示位置，需要在队列头部pop. 所以采用deque双端队列。

截图

R46519188 记录详情

编程语言	代码长度	用时	内存
C++11	1.42KB	2.19s	13.84MB

测试点信息 源代码

测试点信息

#1 AC 4ms/548.00KB	#2 AC 624ms/12.16MB	#3 AC 4ms/448.00KB	#4 AC 5ms/728.00KB	#5 AC 10ms/760.00KB	#6 AC 16ms/800.00KB	#7 AC 66ms/1.74MB
#8 AC 315ms/6.37MB	#9 AC 526ms/12.63MB	#10 AC 625ms/13.84MB				



所属题目 P1886 滑动窗口 / 【模板】单调队列

评测状态 Accepted

评测分数 100

提交时间 2021-02-12 16:27:20

代码

```

#include<iostream>
#include<string>
#include<vector>
#include<queue>
#include<deque>
#include<algorithm>
#include<fstream>
#include <typeinfo>
#include <sstream>
#include <regex>
using namespace std;
struct Node
{
    int val;
    int position;
    Node(int v,int p)
    {
        val = v;
        position = p;
    }
};
class Solution
{
public:
    Solution()
    {
        int n,k;
        cin >> n >> k;
        int * A = new int[n];
        for (int i = 0; i < n; i++)
        {
            cin >> A[i];
        }
        deque<Node> min_que;
        vector<int> min_res;
        deque<Node> max_que;
        vector<int> max_res;
        for (int i = 0; i < n; i++)
        {
            if (i >= k)
            {
                Node head = min_que.front();
                if (head.position < i - k + 1)
                {
                    min_que.pop_front();
                }
                Node head1 = max_que.front();
                if (head1.position < i - k + 1)
                {

```



```

        max_queue.pop_front();
    }
}
while(!min_queue.empty() && A[i] <= min_queue.back().val)
{
    min_queue.pop_back();
}
min_queue.push_back(Node(A[i], i));

while(!max_queue.empty() && A[i] >= max_queue.back().val)
{
    max_queue.pop_back();
}
max_queue.push_back(Node(A[i], i));
if (i >= k-1)
{
    min_res.push_back(min_queue.front().val);
    max_res.push_back(max_queue.front().val);
}

}

for (auto v : min_res)
    cout << v << " ";
cout << endl;
for (auto v : max_res)
    cout << v << " ";
cout << endl;
}
};

int main ()
{
    //ifstream fin("P1886_3.in");
    //cin.rdbuf(fin.rdbuf());
    Solution x = Solution();
    return 0;
}

```

并查集:3367

思路简述

标准的并查集数据结构考察。并查集就是一个可以方便进行集合合并和判断元素属于哪个集合的查找操作的数据结构。我采用了数组A存储，i号元素的父节点是谁A[i]表示i的父节点，如果A[i] < 0 表示i 就是根节点，并且abs(A[i]) 表示该集合的元素个数是多少。

同时在find过程中采用了路径压缩，来提高查询速度。

截图

R45936173 记录详情

编程语言
C++11

代码长度
1.35KB

用时
510ms

内存
624.00KB

测试点信息 源代码

测试点信息

#1 AC 4ms/484.00KB	#2 AC 163ms/604.00KB	#3 AC 4ms/624.00KB	#4 AC 3ms/616.00KB	#5 AC 3ms/448.00KB	#6 AC 4ms/476.00KB	#7 AC 4ms/596.00KB
#8 AC 4ms/608.00KB	#9 AC 162ms/476.00KB	#10 AC 159ms/600.00KB				

所属题目

P3367 【模板】并查集

评测状态

Accepted

评测分数

100

提交时间

2021-02-01 18:16:40

代码

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
#include<fstream>
using namespace std;
class UFsets
{
public:
    UFsets(int N)
    {
        parents = new int[N+1];
        for (int i = 0; i < N+1; i++)
        {
            parents[i] = -1;
        }
    }
    int find(int p) // 不做路径压缩会TLE
    {
        int root = p;
        while(true)
        {
            if (parents[root] < 0)
                break;
            else
                root = parents[root];
        } // 搜索根
        while (p != root)
        {
            int next_p = parents[p];
```

```

        parents[p] = root;
        p = next_p;
    }
    return root;
}

void weightedUnion(int p1, int p2)
{
    int root1, root2;
    root1 = find(p1);
    root2 = find(p2);
    if (root1 != root2)
    {
        int temp = parents[root1] + parents[root2];
        if (parents[root1] > parents[root2])
        {
            parents[root2] = root1;
            parents[root1] = temp;
        }
        else
        {
            parents[root1] = root2;
            parents[root2] = temp;
        }
    }
}

private:
    int *parents;
};

class Solution
{
public:
    Solution()
    {
        int N, M;
        cin >> N>>M;
        UFsets ufsets = UFsets(N);
        int z, x, y;
        for (int i = 0; i < M; i++)
        {
            cin >> z >> x >> y;
            if (z == 1)
            {
                ufsets.weightedUnion(x, y);
            }
            else if (z == 2)
            {
                (ufsets.find(x) == ufsets.find(y)) ? cout << "Y\n" : cout << "N\n";
            }
        }
    }
};

```

```

    }
}
};

int main()
{
    //ifstream fin("in.txt");
    //ofstream fout("out.txt");
    //cin.rdbuf(fin.rdbuf());
    //cout.rdbuf(fout.rdbuf());
    Solution s = Solution();
}

```

康托展开:5367

思路简述

求一个排列的名次，其公式叫做康托展开：如果输入是N,那么结果为

$$A[1] \times (N-1)! + A[2] \times (N-2)! + \dots + A[N-1] \times 1!$$

其中A[i] 表示 排列中第 i 个数 K 减去其前面出现的比 K小的数的个数，也即 A[i] 表示在 第i到第N个数中是第几大。

求阶乘可用前缀积，求A[k] 可用树状数组，树状数组B[k]为0或者1 表示 第k个数是否已经出现。

$$A[i] = \text{sum}(B, i);$$

树状数组的详细介绍见：

<https://www.luogu.com.cn/problem/P3374>

截图

R46576140 记录详情

编程语言	代码长度	用时	内存
C++11	1.80KB	3.71s	19.62MB

测试点信息 源代码

测试点信息

#1 AC 2ms/496.00KB	#2 AC 2ms/624.00KB	#3 AC 3ms/620.00KB	#4 AC 2ms/616.00KB	#5 AC 3ms/496.00KB	#6 AC 3ms/760.00KB	#7 AC 3ms/656.00KB
#8 AC 3ms/624.00KB	#9 AC 2ms/488.00KB	#10 AC 2ms/632.00KB	#11 AC 450ms/19.49MB	#12 AC 255ms/10.73MB	#13 AC 374ms/16.38MB	#14 AC 130ms/5.98MB
#15 AC 401ms/17.11MB	#16 AC 418ms/18.09MB	#17 AC 428ms/18.00MB	#18 AC 323ms/14.00MB	#19 AC 451ms/19.62MB	#20 AC 451ms/19.61MB	



所属题目 P5367 【模板】康托展开

评测状态 Accepted

评测分数 100

提交时间 2021-02-14 18:13:48

代码

```

#include<iostream>
#include<string>
#include<vector>
#include<queue>
#include<deque>
#include<algorithm>
#include<fstream>
#include <typeinfo>
#include <sstream>
#include <regex>
using namespace std;
#define ll long long
class Solution
{
public:
    int N;
    int *a;
    int *tree_array;
    ll *prefix_factor;
    ll mod = 998244353;
    int *lowbit;
    int sum(int index)
    {
        int res = 0;
        while(index > 0)
        {
            res += tree_array[index];
            index -= lowbit[index];
        }
        return res;
    }
    void add(int index,int val)
    {
        while(index <= N)
        {
            tree_array[index] += val;
            index += lowbit[index];
        }
    }
    Solution()
    {
        cin>>this->N;
        this->a = new int[this->N + 1];
        this->tree_array = new int[this->N + 1];
        this->lowbit = new int[this->N+1];
        for(int i=1;i<=this->N;i++)
        {
            cin>>this->a[i];
            lowbit[i]=i&-i;
        }
    }
};

```

```

    }
    for(int i=1;i<=this->N;i++)
    {
        add(i,1);
    }

    prefix_factor = new ll[this->N];
    prefix_factor[1] = 1;
    for(int i=2;i<this->N;i++)
    {
        prefix_factor[i] = (prefix_factor[i-1]*i) % this->mod;
    }
    ll res=0;
    for(int i=1;i<N;i++)
    {
        // cout<<"factor : "<<(N-i)<<" "<<prefix_factor[N-i]<<endl;
        // cout<<"sum: "<<sum(a[i])<<endl;
        res += (prefix_factor[N-i] * (sum(a[i])-1 )) % this->mod;
        add(a[i],-1);
        // for(int i=1;i<=N;i++)
        //     cout<<tree_array[i]<<" ";
        // cout<<endl;
    }
    res+=1;
    cout<<(res % this->mod)<<endl;

}

};

int main ()
{
    //ifstream fin("P1886_3.in");
    //cin.rdbuf(fin.rdbuf());
    Solution x = Solution();
    return 0;
}

```

最长不下降子链 :3902

思路简述

用DP的思想时间复杂度为 $O(n^2)$ 会超时。用二分查找的方法。

截图

R46593367 记录详情

编程语言	代码长度	用时	内存
C++11	632B	158ms	996.00KB

测试点信息 源代码

测试点信息

#1 AC 2ms/484.00KB	#2 AC 3ms/616.00KB	#3 AC 3ms/548.00KB	#4 AC 3ms/628.00KB	#5 AC 2ms/616.00KB	#6 AC 25ms/872.00KB	#7 AC 39ms/996.00KB
#8 AC 23ms/632.00KB	#9 AC 24ms/692.00KB	#10 AC 34ms/908.00KB				



所属题目	P3902 递增
评测状态	Accepted
评测分数	100
提交时间	2021-02-15 11:16:43

代码

```
#include<iostream>
#include<string>
#include<vector>
#include<queue>
#include<deque>
#include<algorithm>
#include<fstream>
#include <typeinfo>
#include <sstream>
#include <regex>
using namespace std;
#define ll long long
int main()
{
    int n;
    int seq_len = 0;
    cin>>n;
    int *seq = new int[n+1];
    memset(seq,0,sizeof(int)*(n+1));
    for(int i=1;i<=n;i++)
    {
        int cur_val;
        cin>>cur_val;
        if(cur_val>seq[seq_len])
            seq[++seq_len]=cur_val;
        else
        {
            *lower_bound(seq+1,seq+seq_len+1,cur_val)=cur_val;
        }
    }
    cout<<n-seq_len<<endl;
    return 0;
}
```

背包问题:1616

思路简述

完全背包问题, $dp[i][j]$ 表示前*i*个物品在*j*背包内所能产生的最大价值。

状态转移方程 $dp[i][j] = \max(dp[i-1][j], dp[i][j-w[i]] + v[i])$; 但是会空间超限。空间优化方法, 发现 $dp[i][j]$ 只依赖这一行前面的数和上一行的列相同的一个数。所以只保留一行就行。

$$dp[j] = \max(dp[j], dp[j-w[i]] + v[i])$$

截图

R46590336 记录详情

编程语言
C++11

代码长度
1.25KB

用时
126ms


内存
76.98MB

测试点信息

源代码

测试点信息

#1 AC 2ms/496.00KB	#2 AC 2ms/632.00KB	#3 AC 2ms/508.00KB	#4 AC 7ms/736.00KB	#5 AC 8ms/640.00KB	#6 AC 6ms/752.00KB	#7 AC 5ms/616.00KB
#8 AC 5ms/744.00KB	#9 AC 26ms/29.52MB	#10 AC 63ms/76.98MB				

 cz_nju

所属题目
P1616 疯狂的采药

评测状态
Accepted

评测分数
100

提交时间
2021-02-15 09:44:30

代码

```
#include<iostream>
#include<string>
#include<vector>
#include<queue>
#include<deque>
#include<algorithm>
#include<fstream>
#include <typeinfo>
#include <sstream>
#include <regex>
using namespace std;
#define ll long long
const int N=1e4+5;
ll w[N],v[N];
int main ()
{
    // ifstream fin("P1616_1.in");
    // cin.rdbuf(fin.rdbuf());
    ll t,m;
    cin>>t>>m;
    for(int i=1;i<=m;i++)
```



```

{
    cin>>w[i]>>v[i];
}
//二维会空间超限。 看看动态转移方差 dp[i][j] 依赖于上一行的dp[i-1][j] 和当前这一行
dp[i][j-w[i]]
//所以说只要保留当前一行就行了
//上一行的保留在原来的位置
//这一行从前往后更新就可
ll *dp = new ll[t+1];
memset(dp,0,sizeof(ll)*(t+1));
for(int i=1;i<=m;i++)
{
    for(int j=w[i];j<=t;j++)
    {
        dp[j] = max(dp[j],dp[j-w[i]]+v[i]);
    }
}
cout<<dp[t]<<endl;
return 0;
}

```

NIM博弈:2197

思路简述

参考资料: https://blog.csdn.net/A_Comme_Amour/article/details/79347291

经典问题，只需证明所有石子异或和为0则先手必输。证明思路见参考资料。

截图

R46558684 记录详情

编程语言	代码长度	用时	内存
C++11	768B	69ms	620.00KB

测试点信息
源代码

测试点信息

#1	#2	#3	#4	#5
AC	AC	AC	AC	AC
14ms/476.00KB	9ms/620.00KB	15ms/620.00KB	12ms/612.00KB	19ms/488.00KB

所属题目
P2197 【模板】nim游戏

评测状态
Accepted

评测分数
100

提交时间
2021-02-14 10:07:56

代码

```

#include<iostream>
#include<string>
#include<vector>
#include<queue>

```

```

#include<deque>
#include<algorithm>
#include<fstream>
#include <typeinfo>
#include <sstream>
#include <regex>
using namespace std;

class Solution
{
public:
    Solution()
    {
        int T;
        cin>>T;
        while(T--)
        {
            int n;
            cin>>n;
            int ans=0;
            cin>>ans;
            for(int i=1;i<n;i++)
            {
                int temp;
                cin>>temp;
                ans ^= temp;
            }
            if(ans)
                cout<<"Yes"<<endl;
            else
                cout<<"No"<<endl;
        }
    }
};

int main ()
{
    //ifstream fin("P1886_3.in");
    //cin.rdbuf(fin.rdbuf());
    Solution x = Solution();
    return 0;
}

```