Hihocoder 题目泛做解题报告

RejudgeX @ SJTU

2016 年 10 月 4 日

目录

1 滚粗日记

1.1 1001 : A+B

Default , Hello World :)

```
#include<cstdio>
using namespace std;

int main(){
    int a, b;
    //this is a very very typical problem
    //I have ever sloved, so I am proud of myself
    while(~scanf("%d%d", &a, &b)){
        printf("%d\n", a+b);
    }
    return 0;
}
```

1.2 1014 : Trie 树

- 经典字典树 (Trie)
- Trie 是一种基于字符串前缀保存的数据结构，将每个字符串以一棵树的形式保存，而相同的前缀会在同一颗树的路径上.
- 关于 Trie 常见的操作有：Build, Update, Query. Update | Query 的时候由于仅仅按照字符串的长度进行逐位操作，所以复杂度为 $O(d)$，build 的复杂度为 $O(n * d)$.
- Trie 的扩展有基于 Trie 的贪心 Xor(经典问题如：查找 N 个数最大的两个异或值，查找区间最大的异或和等).

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
char s[20];

typedef struct Trie{
    int val, cnt;
    Trie *next[26];
    Trie(){
        for(int i = 0; i < 26; ++i){
            next[i] = NULL;
        }
    }
```

```
}Trie;

Trie* buildTrieTree(Trie *root, char *dict){
    Trie *p = root, *q;
    int i = 0;
    while(dict[i] != '0'){
        int k = dict[i] - 'a';
        if(p->next[k] == NULL){
            q = new Trie();
       //    for(int j = 0; j < 26; ++j){
       //        q->next[j] = NULL;
       //    }
            q->val = k;
            q->cnt = 0;
            p->next[k] = q;
        }
        p = p->next[k];
        p->cnt += 1;
        ++i;
    }
    return root;
}

int query(Trie *root, char *dict){
    int p = 0;
    while(dict[p] != '0'){
        int k = dict[p] - 'a';
        if(root->next[k] != NULL){
            root = root->next[k];
        }
        else{
            return 0;
        }
        ++p;
    }
    return root->cnt;
}

int main(){
    int n, m;

    scanf("%d", &n);
    Trie *root = new Trie();
  // for(int i = 0; i < 26; ++i){
  //     root->next[i] = NULL;
  // }
```

```
    for(int i = 0; i < n; ++i){
        scanf("%s", s);
        root = buildTrieTree(root, s);
    }

    scanf("%d", &m);
    for(int i = 0; i < m; ++i){
        scanf("%s", s);
        int ans = query(root, s);
        printf("%d\n", ans);
    }
    return 0;
}
```

1.3 1015：KMP 算法

- 经典 KMP
- KMP 太经典了, 关于匹配串与模式串的匹配问题，利用的是最长前缀与后缀相等来减少不必要的匹配，将匹配的复杂度从 $O(n*m)$ 降到 $O(n+m)$.
- KMP 的 next 数组可以做很多扩展，最常见的是求循环节。注意理解 next 数组的本质，本质上 next 是多次迭代的，而每次迭代始终保持前缀与后缀相等，这个性质同样重要。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N = 1e4 + 1;
char a[N], b[N*100];
int p[N];

void GetNextArray(int n, char *b){
    int i, j;
    p[0] = -1;
    j = -1;
    for(i = 1; i < n; ++i){
        while(j >= 0 && b[j+1] != b[i]){
            j = p[j];
        }
        if(b[j+1] == b[i]){
            ++j;
        }
        p[i] = j;
    }
```

```
}

void KMP(int n1, int n2, char *a, char *b){
    int j = -1, ans = 0;
    for(int i = 0; i < n1; ++i){
        while(j >= 0 && b[j+1] != a[i]){
            j = p[j];
        }
        if(b[j+1] == a[i]){
            ++j;
        }
        if(j == n2-1){
            ++ans;
            j = p[j];
        }
    }
    printf("%d\n", ans);
}

int main(){
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; ++i){
        scanf("%s%s", a, b);
        int n1 = strlen(a), n2 = strlen(b);
        GetNextArray(n1, a);
        KMP(n2, n1, b, a);
    }
    return 0;
}
```

1.4 1032：最长回文子串

- 经典 Manacher 求最长回文串
- 字符串最长回文串最简单的就是 $O(n^2)$ 的暴力匹配 (枚举中心点), 当然也可以求正向串与逆向串的最长公共子序列, 也是 $O(n^2)$. 也可以通过枚举 i 点的前缀与后缀，基于后缀数组与高度数组, 求后缀与前缀的最长公共前缀 (LCP) 来解决，但是复杂度与效率仍然不及 Manacher.
- Manacher 的算法细节不表,说一下大概思路：在遍历字符串的过程中使用记录两个变量值 mx, idx 分别表示迭代到当前位置，最远的回文串能达到的位置以及其对应回文中心的下标。考虑对称性, 迭代到 i 的时候，考虑 i 关于 idx 的对称点 j，这样通过判断可以将 j 的回文长度加入到 i 的回文串中，这部分计算之前已经产生，所以节省了大量重复计算。最终比较一下所有位置的回文串长度即可。

- 算法的复杂度为 $O(n)$。有很多细节,比如预处理, 位置比较等等,可以参考：_https://www.felix021.com/blog/read.

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
#define minab(a, b) ((a) < (b) ? (a) : (b))
const int N = 1000010;

int p[N<<1];
char s[N<<1];
string st;
int mx, idx;//the Rightest postion can be reached now, and the idx

void manacher(char *str) {
    mx = 0, idx = 0;
    int ans = 0;
    memset(p, 0, sizeof(p));
    int size = (int)strlen(str);

    for(int i = 1; str[i]; ++i) {
        p[i] = mx > i ? minab(p[idx*2 - i], mx - i) : 1;
        while((p[i] + i < size) && (i - p[i] >= 0) && str[p[i]+i] == str[i-p[i]]) {
            ++p[i];
        }

        if(i + p[i] > mx) {
            mx = i + p[i];
            idx = i;
        }
    //   printf("p[i] = %d\n", p[i]);
        if(str[i] == '#') {
            if(((p[i]) / 2) * 2 > ans) {
                ans = (p[i]) / 2 * 2;
            }
        }
        else {
            if((p[i] - 1)/2 * 2 + 1 > ans) {
                ans = (p[i] - 1) / 2 * 2 + 1;
            }
        }
    }
    printf("%d\n", ans);
}
```

```
int main() {
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; ++i) {
        cin >> st;
        for(int j = 0; j < st.size(); ++j) {
            s[j<<1|1] = '#';
            s[(j<<1) + 2] = st[j];
        }
        s[st.size() << 1 | 1] = '#';
        s[(st.size() << 1) + 2] = '0';
     //  printf("%s\n", s+1);
        manacher(s+1);
    }
    return 0;
}
```

1.5 1033：交错和

- 数位 DP
- 数位 DP 是一类按位进行 DP 统计的问题。按位统计之后，发现有很多重复子问题，于是就可以愉快的 DP 了.
- 数位 DP 的写法比较固定，一般都是用记忆化搜索的写法 (参考我的代码. 注意需要考虑前导 0, 以及每次枚举的数字的范围 (能不能到 9).
- 对于本题 dp[len][sum] 记录的是长度为 len 的数字序列交错和能达到 sum 的情况有多少种. 复杂度 $O(d * k)$

```
// Author:tcOops
// Level -> CF/TC -> Yellow
// > -> Ag
// -> F/L/A/G
// -> Latency 「2017/5/15」

#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 21;
const int MOD = 1e9 + 7;
const int OFFSET = 210;

int dig[N];
```

```
long long base[N];

struct google {
    long long n, sum;
    google() {
        n = -1, sum = 0;
    }
};
google dp[N][420];


google solve(int len, long long sum, bool isPrefix0, bool limit) {
    //printf("%d %d %lld\n", len, digital, sum);
    google ans, res;
    ans.n = 0, ans.sum = 0;

    if(len == 0) {
        if(sum == 0) {
            ans.n = 1;
        }
        return ans;
    }

    if(!limit && isPrefix0 && dp[len][sum+OFFSET].n != -1) {
        return dp[len][sum+OFFSET];
    }

    int up = limit ? dig[len] : 9;

    for(int i = 0; i <= up; ++i) {
        if(!isPrefix0) {
            if(i == 0) {
                res = solve(len-1, sum, false, limit && (i == up));
            }
            else {
                res = solve(len-1, i-sum, true, limit && (i == up));
            }
        }
        else {
            long long new_sum = i - sum;
            res = solve(len-1, new_sum, true, limit && (i == up));
        }

        ans.n += res.n;
        ans.sum = ((ans.sum + res.sum) % MOD + (res.n*i) % MOD * base[len-1] % MOD + MOD) % MOD;
    }
```

```
    //printf("%lld %lld\n", ans.n, ans.sum);
    if(!limit && isPrefix0) {
        dp[len][sum+OFFSET] = ans;
    }
    return ans;
}


long long go(long long n, long long k) {
    if(n <= 0) {
        return 0;
    }

    int len = 0;
    while(n) {
        dig[++len] = n%10;
        n /= 10;
    }
   // dig[len++] = 0;
    return solve(len, k, false, true).sum;
}


int main () {
    long long l, r, k;
    base[0] = 1;
    for(int i = 1; i < N; ++i) {
        base[i] = (base[i-1] * 10) % MOD;
    }

    scanf("%lld %lld %lld", &l, &r, &k);
    printf("%lld\n", (go(r, k) - go(l-1, k) + MOD) % MOD);
    return 0;
}
```

1.6 1039：字符消除

- 枚举、模拟
- 考虑到字符串 s 的长度不超过 100，完全可以暴力枚举解决。
- 枚举可以替换的位置，将该位置分别填充为'A','B','C' 进行消除，消除的过程简单模拟即可. 复杂度 $O(n^2)$

```cpp
//AC

// Author: RejudgeX
// Level -> CF/TC -> Yellow
// > -> Ag
// -> F/L/A/G
// -> Latency 「2017/5/15」


#include <iostream>
#include <cmath>
#include <cstring>
#include <string>
#include <cstdio>
#include <set>
#include <algorithm>
#include <queue>
#include <vector>
#include <map>
#include <ctime>
//#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
#define fi first
#define se second
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll MOD = 1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=MOD;for(;b;b>>=1){if(b&1)res=res*a%MOD;a=a*a%MOD;}return res;}
// head

inline void gn(long long &x){
    int sg=1;
    char c;while(((c=getchar())<'0'||c>'9')&&c!='-');c=='-'?(sg=-1,x=0):(x=c-'0');
    while((c=getchar())>='0'&&c<='9')x=x*10+c-'0';x*=sg;
}

inline void gn(int&x){long long t;gn(t);x=t;}
inline void gn(unsigned long long&x){long long t;gn(t);x=t;}
```

```
inline void gn(double&x){double t;scanf("%lf",&t);x=t;}
inline void gn(long double&x){double t;scanf("%lf",&t);x=t;}

string change[3] = {"A", "B", "C"};

int solve(string word) {
  int ans = 0;
  for(int i = 0; i < word.size(); ++i) {
    for(int t = 0; t < 3; ++t) {
      string word1 = word.substr(0, i+1) + change[t] + word.substr(i+1);
      int cnt = 0;
      while(true) {
        string word2 = "";
        bool suc = false;
        for(int j = 0; j < word1.size();) {
            int k = j;
            while(k < word1.size() && word1[k] == word1[j]) {
              ++k;
            }
            if(k - j > 1) {
              suc = true;
              cnt += k - j;
            }
            else {
              word2 += word1[j];
            }
            j = k;
        }
        if(!suc) {
          break;
        }
        word1 = word2;
      }
      if(cnt > ans) {
        ans = cnt;
      }
    }
  }
  return ans;
}

int main() {
  int n;gn(n);
  for(int i = 0; i < n; ++i) {
    string word;
    cin >> word;
```

```
    cout << solve(word) << endl;
  }
  return 0;
}
```

1.7 1040 : 矩形判断

- 模拟题
- 考虑围成矩形的条件：(1) 四条边首尾相连，能形成一个闭合的环。(2) 这个环中的 (1,3)(2,4) 边互相平行.
- dfs4 条边所有的全排列，check 是否满足上面两个条件. 时间复杂度 $O(1)$

```
// Author: RejudgeX
// Level -> CF/TC -> Yellow
// > -> Ag
// -> F/L/A/G
// -> Latency 「2017/5/15」

//Google mock-contest 1
//RejudgeX: 2016/8/3 22:35 - 24:00
#include <iostream>
#include <cmath>
#include <cstring>
#include <string>
#include <cstdio>
#include <set>
#include <algorithm>
#include <queue>
#include <vector>
#include <map>
#include <ctime>
//#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
#define fi first
#define se second
typedef vector<int> VI;
typedef long long ll;
```

```
typedef pair<int,int> PII;
const ll MOD = 1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=MOD;for(;b;b>>=1){if(b&1)res=res*a%MOD;a=a*a%MOD;}return res;}
// head

inline void gn(long long &x){
    int sg=1;
    char c;while(((c=getchar())<'0'||c>'9')&&c!='-');c=='-'?(sg=-1,x=0):(x=c-'0');
    while((c=getchar())>='0'&&c<='9')x=x*10+c-'0';x*=sg;
}

inline void gn(int&x){long long t;gn(t);x=t;}
inline void gn(unsigned long long&x){long long t;gn(t);x=t;}
inline void gn(double&x){double t;scanf("%lf",&t);x=t;}
inline void gn(long double&x){double t;scanf("%lf",&t);x=t;}

int g[5][5];
int res[10], node[4];
bool vis[5];

struct seg {
  int x1, y1, x2, y2;
}segs[4];
bool suc;

bool joint() {
  for(int i = 1; i < 8; i += 2) {
    int x1, y1, x2, y2;
    if(res[i] & 1) {
      x1 = segs[res[i] >> 1].x2;
      y1 = segs[res[i] >> 1].y2;
    }
    else {
      x1 = segs[res[i] >> 1].x1;
      y1 = segs[res[i] >> 1].y1;
    }

    if(res[(i+1)%8] & 1) {
      x2 = segs[res[(i+1)%8] >> 1].x2;
      y2 = segs[res[(i+1)%8] >> 1].y2;
    }
    else {
      x2 = segs[res[(i+1)%8] >> 1].x1;
      y2 = segs[res[(i+1)%8] >> 1].y1;
    }
    if(x1 != x2 || y1 != y2) {
```

```
      return false;
    }
    return true;
  }
}

bool fits() {
  int dx1 = segs[node[0]].x2 - segs[node[0]].x1;
  int dy1 = segs[node[0]].y2 - segs[node[0]].y1;
  int dx2 = segs[node[1]].x2 - segs[node[1]].x1;
  int dy2 = segs[node[1]].y2 - segs[node[1]].y1;
  int dx3 = segs[node[2]].x2 - segs[node[2]].x1;
  int dy3 = segs[node[2]].y2 - segs[node[2]].y1;
  int dx4 = segs[node[3]].x2 - segs[node[3]].x1;
  int dy4 = segs[node[3]].y2 - segs[node[3]].y1;

  if((!dx1 && !dy1) || (!dx2 && !dy2) || (!dx3 && !dy3) || (!dx4 && !dy4)) {
    return false;
  }

  if(dx1 * dy3 != dx3 * dy1) {
    return false;
  }
  if(dx2 * dy4 != dx4 * dy2) {
    return false;
  }

  if(dx1 * dx2 + dy1 * dy2 != 0) {
    return false;
  }
  if(dx2 * dx3 + dy2 * dy3 != 0) {
    return false;
  }
  if(dx3 * dx4 + dy3 * dy4 != 0) {
    return false;
  }
  if(dx4 * dx1 + dy4 * dy1 != 0) {
    return false;
  }
  return true;
}

bool dfs(int k) {
  if(k == 4) {
    if(joint() && fits()) {
      printf("YES\n");suc = true;
```

```
      return true;
    }
  }
  for(int v = 0; v < 4; ++v) {
    if(!vis[v]) {
      res[k<<1] = v<<1; res[k<<1|1] = v<<1|1; vis[v] = true;
      node[k] = v;
      if(dfs(k+1)) {
        return true;
      }
      res[k<<1] = v<<1|1; res[k<<1|1] = v<<1;
      if(dfs(k+1)) {
        return true;
      }
      vis[v] = false;
    }
  }
  return false;
}

int main() {
  int T;
  gn(T);
  while(T--) {
    for(int i = 0; i < 4; ++i) {
      gn(segs[i].x1); gn(segs[i].y1);
      gn(segs[i].x2); gn(segs[i].y2);
    }
    memset(vis, false, sizeof(vis));
    suc = false;
    dfs(0);
    if(!suc) {
      printf("NO\n");
    }
  }
  return 0;
}
```

1.8 1041：国庆出游

- DFS 标记、BitSet
- 题目的操作对象是一颗树, 对于树上的部分节点有先后遍历顺序要求，现在要访问所有节点一次, 且树上的每条边都只访问两次（来回各一次
- 从部分节点的遍历的顺序要求入手，只要能满足这部分节点的遍历顺序就一定找到解. 在树上遍

历当然是 dfs 比较方便,dfs 到当前点需要考虑当前需要的次序是否在这个点的子树下面，如果在就 dfs 下去，且标记过程中所有的边，如果全都找不到就说明无解。

- 所以可以先 dfs 预处理一遍，找出节点 u 的子树包含的节点集合这样就能在 O(1) 时间进行判断. 考虑到 n 不大，直接用 bitset 搞定.

```
//AC

// Author: RejudgeX
// Level -> CF/TC -> Yellow
// > -> Ag
// -> F/L/A/G
// -> Latency 「2017/5/15」


#include <iostream>
#include <cmath>
#include <cstring>
#include <string>
#include <cstdio>
#include <set>
#include <algorithm>
#include <queue>
#include <vector>
#include <map>
#include <ctime>
#include <bitset>
//#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
#define fi first
#define se second
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll MOD = 1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=MOD;for(;b;b>>=1){if(b&1)res=res*a%MOD;a=a*a%MOD;}return res;}
// head

inline void gn(long long &x){
    int sg=1;
```

```
    char c;while(((c=getchar())<'0'||c>'9')&&c!='-');c=='-'?(sg=-1,x=0):(x=c-'0');
    while((c=getchar())>='0'&&c<='9')x=x*10+c-'0';x*=sg;
}

inline void gn(int&x){long long t;gn(t);x=t;}
inline void gn(unsigned long long&x){long long t;gn(t);x=t;}
inline void gn(double&x){double t;scanf("%lf",&t);x=t;}
inline void gn(long double&x){double t;scanf("%lf",&t);x=t;}

const int N = 110;
vector<int> g[N];
bitset<N > f[N];
bool used[N][N];
int a[N], n, m, idx;
bool suc;

void pre(int u, int fa) {
  f[u][u] = 1;
  for(auto v : g[u]) {
    if(v == fa) continue;
    pre(v, u);
    f[u] |= f[v];
  }
}

void solve(int u, int fa) {
  if(idx < m && a[idx] == u) {
    ++idx;
  }
  if(idx == m) {
    suc = true;
    return ;
  }

  while(idx < m) {
    int res = idx;
    int need = a[idx];
    for(auto v : g[u]) {
      if(v == fa) continue;
      if(f[v][need] && !used[u][v]) {
        used[u][v] = true;
        solve(v, u);
        break;
      }
    }
    if(res == idx) {
```

```
      break;
    }
  }
}

int main() {
  int T; gn(T);
  while(T--) {
    gn(n);
    memset(used, true, sizeof(used));
    suc = false;
    for (int i = 1 ; i <= n ; i ++) {
         g[i].clear();
         f[i].reset();
    }
    for(int i = 1; i < n; ++i) {
      int x, y;
      gn(x); gn(y);
      used[x][y] = used[y][x] = false;
      g[x].push_back(y), g[y].push_back(x);
    }
    gn(m);
    for(int i = 0; i < m; ++i) {
      gn(a[i]);
    }
    pre(1, 0);

    idx = 0;
    solve(1, 0);
    if(suc) {
      cout << "YES" << endl;
    }
    else {
      cout << "NO" << endl;
    }
  }
  return 0;
}
```

1.9 1042：跑马圈地

- 大模拟
- 首先这题的正解我并不知道是什么，我 AC 的算法基于：最后圈出的一定是一个矩形。
- 基于这一点，我分类大讨论了矩形与水塘的位置关系所有情况。对于每种情况，枚举圈出的矩形

的长宽，然后判断需要的周长是否不超过 L，如果是就统计面积，进行比较.

- 总之就是个大模拟辣 T_T, 复杂度 $O(nm(n+m))$?

```
//AC

// Author: RejudgeX
// Level -> CF/TC -> Yellow
// > -> Ag
// -> F/L/A/G
// -> Latency 「2017/5/15」

#include <iostream>
#include <cmath>
#include <cstring>
#include <string>
#include <cstdio>
#include <set>
#include <algorithm>
#include <queue>
#include <vector>
#include <map>
#include <ctime>
#include <bitset>
//#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
#define fi first
#define se second
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll MOD = 1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=MOD;for(;b;b>>=1){if(b&1)res=res*a%MOD;a=a*a%MOD;}return res;}
// head

inline void gn(long long &x){
    int sg=1;
    char c;while(((c=getchar())<'0'||c>'9')&&c!='-');c=='-'?(sg=-1,x=0):(x=c-'0');
    while((c=getchar())>='0'&&c<='9')x=x*10+c-'0';x*=sg;
}
```

```
inline void gn(int&x){long long t;gn(t);x=t;}
inline void gn(unsigned long long&x){long long t;gn(t);x=t;}
inline void gn(double&x){double t;scanf("%lf",&t);x=t;}
inline void gn(long double&x){double t;scanf("%lf",&t);x=t;}

int n, m, L, l, r, t, b;

int getCircle(int ll, int rr, int bb, int tt) {
  //inside
  if(bb >= b && tt <= t && ll >= l && rr <= r) {
  // printf("lll\n");
    return -1;
  }

  if(tt >= t && bb >= b && bb <= t && ll >= l && rr <= r) {
    int cir = (tt - t + rr - ll) * 2;
    if(cir <= L) {
      return (tt - t) * (rr - ll);
    }
    else {
      return -1;
    }
  }

  if(ll <= l && rr >= l && rr <= r && tt <= t && bb >= b) {
    int cir = (tt - bb + l - ll) * 2;
    if(cir <= L) {
      return (tt - bb) * (l - ll);
    }
    return -1;
  }

  if(bb <= b && tt >= b && tt <= t && ll >= l && rr <= r) {
    int cir = (b - tt + rr - ll) * 2;
    if(cir <= L) {
      return (b - tt) * (rr - ll);
    }
    else {
      return -1;
    }
  }
  if(rr >= r && ll <= r && ll >= l && tt <= t && bb >= b) {
    int cir = (rr - r + tt - bb) * 2;
    if(cir <= L) {
      return (tt - bb) * (rr - r);
```

```
  }
  else {
    return -1;
  }
}
//0 point
if(bb >= t || tt <= b || ll >= r || rr <= l) {
  //cout << "aaa" << endl;
  if(2*(rr - ll + tt - bb) <= L) {
    return (rr - ll) * (tt - bb);
  }
  else {
    return -1;
  }
}

// one point in
if(tt >= t && bb <= t && bb >= b && ll <= l && rr >= l && rr <= r) {
  //cout << "bbb" << endl;
  int cir = 2 * (rr - ll + tt - bb);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (t - bb) * (rr - l);
  }
  else {
    return -1;
  }
}
if(tt >= t && bb <= t && bb >= b && ll >= l && ll <= r && rr >= r) {
  //cout << "ccc" << endl;
  int cir = 2 * (rr - ll + tt - bb);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (t - bb) * (r - ll);
  }
  else {
    return -1;
  }
}
if(tt <= t && tt >= b && bb <= b && ll <= l && rr >= l && rr <= r) {
  //cout << "ddd" << endl;
  int cir = 2 * (rr - ll + tt - bb);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (tt - b) * (rr - l);
  }
  else {
    return -1;
  }
```

```
}
if(tt <= t && tt >= b && bb <= b && ll >= l && ll <= r && rr >= r) {
  //cout << "eee" << endl;
  int cir = 2 * (rr - ll + tt - bb);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (tt - b) * (r - ll);
  }
  else {
    return -1;
  }
}

//two points in
if(tt >= t && ll <= b && ll <= l && rr >= l && rr <= r) {
  //cout << "fff" << endl;
  int cir = 2 * (rr - ll + tt - bb) + 2 * (rr - l);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (rr - l) * (t - b);
  }
  else {
    return -1;
  }
}
if(tt >= t && ll <= b && ll >= l && ll <= r && rr >= r) {
  //cout << "hhh" << endl;
  int cir = 2 * (rr - ll + tt - bb) + 2 * (r - ll);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (r - ll) * (t - b);
  }
  else {
    return -1;
  }
}
if(ll <= l && rr >= r && tt >= t && bb <= t && bb >= b) {
  //cout << "iii" << endl;
  int cir = 2 * (rr - ll + tt - bb) + 2 * (t - bb);
  if(cir <= L) {
    return (rr - ll) * (tt - bb) - (t - bb) * (r - l);
  }
  else {
    return -1;
  }
}
if(ll <= l && rr >= r && tt <= t && tt >= b && bb <= b) {
  //cout << "jjj" << endl;
  int cir = 2 * (rr - ll + tt - bb) + 2 * (tt - b);
```

```
    if(cir <= L) {
      return (rr - ll) * (tt - bb) - (tt - b) * (r - l);
    }
    else {
      return -1;
    }
  }

  //four points in
  if(tt >= t && bb <= b && ll <= l && rr >= r) {
    //cout << "kkk" << endl;
    int cir = 2 * (rr - ll + tt - bb);
    if(cir <= L) {
      return (rr - ll) * (tt - bb) - (r - l) * (t - b);
    }
    else {
      return -1;
    }
  }
  return -1;
}

int main() {
  gn(n); gn(m); gn(L);
  gn(l); gn(r); gn(b); gn(t);

  int ans = 0;
  for(int i = 0; i <= n; ++i) {
    for(int j = i + 1; j <= n; ++j) {
      for(int p = 0; p <= m; ++p) {
        for(int q = p + 1; q <= m; ++q) {
          int mj = getCircle(p, q, i, j);
          if(mj > ans) {
            ans = mj;
          }
        }
      }
    }
  }
  cout << ans << endl;
  return 0;
}
```

1.10 1044：状态压缩·一

- 状态压缩 DP

```
// Author:tcOops
// Level -> CF/TC -> Yellow
// > -> Ag
// -> F/L/A/G
// -> Latency 「2017/5/15」

#include <iostream>
#include <cmath>
#include <cstring>
#include <cstdio>
#include <set>
#include <algorithm>
using namespace std;
#define maxab(a, b) ((a) > (b) ? (a) : (b))
const int N = 1010;

int cab[N];
int dp[N][1<<11];

int cal(int s, int m) {
    int ret = 0;
    for(int i = 0; i < m; ++i) {
        if(s & (1<<i)) {
            ++ret;
        }
    }
    return ret;
}

int main() {

    int n, m, q;
    scanf("%d %d %d", &n, &m, &q);
    for(int i = 1; i <= n; ++i) {
        scanf("%d", &cab[i]);
    }

    memset(dp, 0, sizeof(dp));
   // dp[1][1] = cab[1];
    for(int i = 0; i < n; ++i) {
        for(int s = 0; s < (1 << (m-1)); ++s) {
            int bits = cal(s, m-1);
```

```
            if(bits > q) {
                continue;
            }
            if(bits < q) {
                dp[i+1][s>>1 | (1<<(m-2))] = maxab(dp[i][s] + cab[i+1], dp[i+1][s>>1 | (1<<(m-2))]);
            }
            dp[i+1][s>>1] = maxab(dp[i][s], dp[i+1][s>>1]);
        }
    }

    int ans = 0;
    for(int s = 0; s < (1<<(m-1)); ++s) {
        if(dp[n][s] > ans) {
            ans = dp[n][s];
        }
    }
    printf("%d\n", ans);
    return 0;
}
```