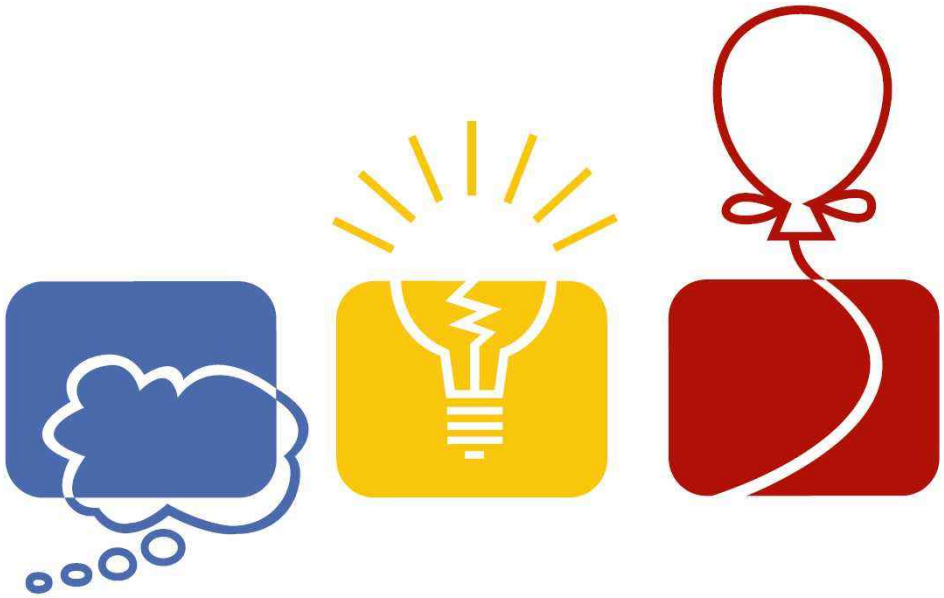

ACM/ICPC Template Manaual

ZheJiang GongShang University

Happy Otaku

February 16, 2019

Contents

0 头文件

```cpp
// 巨菜的ACMer-Happy233

#include <bits/stdc++.h>

using namespace std;

//-----
typedef double db;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pw(x) (1ll << (x))
#define sz(x) ((int)(x).size())
#define all(x) (x).begin(),(x).end()
#define rep(i, l, r) for(int i=(l);i<(r);++i)
#define per(i, l, r) for(int i=(r)-1;i>=(l);--i)
#define sf(x) scanf("%d", &(x))

const double pi = acos(-1);
```

1 字串符

1.1 KMP

```
template<class elemType>
inline void NEXT(elemType &T, vector<int> &next) {
    next[0] = -1;
    for (int i = 1; i < T.size(); i++) {
        int j = next[i - 1];
        while (j >= 0 && T[i - 1] != T[j]) j = next[j];
        if (j >= 0 && T[i - 1] == T[j]) next[i] = j + 1;
        else next[i] = 0;
    }
}

template<class elemType>
inline int COUNT_KMP(elemType &S, elemType &T) {
    vector<int> next(T.size());
    NEXT(T, next);
    int index, count = 0;
    for (index = 0; index < S.size(); ++index) {
        int pos = 0;
        int iter = index;
        while (pos < T.size() && iter < S.size()) {
            if (S[iter] == T[pos]) {
                ++iter;
                ++pos;
            } else {
                if (pos == 0) ++iter;
                else pos = next[pos - 1] + 1;
            }
        }
        if (pos == T.size() && (iter - index) == T.size()) ++count;
    }
    return count;
}

template<class elemType>
inline void NEXT(elemType T[], int count, vector<int> &next) {
    next[0] = -1;
    for (int i = 1; i < count; i++) {
        int j = next[i - 1];
        while (j >= 0 && T[i - 1] != T[j]) j = next[j];
        if (j >= 0 && T[i - 1] == T[j]) next[i] = j + 1;
        else next[i] = 0;
    }
}

template<class elemType>
inline int COUNT_KMP(elemType S[], int c1, elemType T[], int c2) {
    vector<int> next(c2);
    NEXT(T, c2, next);
    int index, count = 0;
    for (index = 0; index < c1; ++index) {
        int pos = 0;
        int iter = index;
        while (pos < c2 && iter < c1) {
            if (S[iter] == T[pos]) {
                ++iter;
```

```
56                 ++pos;
57             }
58             else {
59                 if (pos == 0) ++iter;
60                 else pos = next[pos - 1] + 1;
61             }
62         }
63         if (pos == c2 && (iter - index) == c2) ++count;
64     }
65     return count;
66 }
```

2 动态规划

2.1 01Bag

```
void dp(int n, int m) {
    // n=物品个数
    for (int i = 0; i < n; i++) {
        // m=背包最大容量
        for (int j = m; j >= wei[i]; j--)
            // wei=大小 val=价值
            f[j] = max(f[j], f[j - wei[i]] + val[i]);
    }
}
```

2.2 BagProblem

```
#define N 1000
// val=价值 wei=重量 num=数量
int val[N], wei[N], num[N], f[N];
// n=种类个数 m=背包最大值

// 01背包
void dp1(int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = m; j >= wei[i]; j--)
            f[j] = max(f[j], f[j - wei[i]] + val[i]);
    }
}

// 完全背包
void dp2(int n, int m) {
    //初始化看要求
    for (int i = 0; i <= m; i++) {
        f[i] = INF;
    }
    f[0] = 0;
    //若要求恰好装满背包，那在初始化时除了f[0]=0其它f[1..V]均=-∞
    //若没要求背包装满，只希望价格大，初始化时应将f[0..V]=0)
    for (int i = 0; i < n; i++)
        for (int j = wei[i]; j <= m; j++)
            f[j] = max(f[j], f[j - wei[i]] + val[i]);
}

// 多重背包
void dp3(int n, int m) {
    for (int i = 0; i < n; i++)
        for (int k = 0; k < num[i]; k++)
            for (int j = m; j >= wei[i]; j--)
                f[j] = max(f[j], f[j - wei[i]] + val[i]);
}
```

2.3 FullBag

```
/*
完全背包问题的特点是，每种物品可以无限制的重复使用，可以选择放或不放。
完全背包问题描述:
有N物品和一个容量为V的背包。第i件物品的重量是wei[i]，价值是val[i]。
```

```
*/

#include <cstdio>
#define INF 0x3fffffff
#define N 10047
int f[N],val[N],wei[N];
int min(int a,int b)
{
    return x<y?x:y;
}
int main()
{
    int t,i,j,k,E,F,m,n;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d",&E,&F);
        int c = F-E;
        for(i = 0 ; i <= c ; i++)
            f[i]=INF;
        scanf("%d",&n);
        for(i = 0 ; i < n ; i++)
        {
            scanf("%d%d",&val[i],&wei[i]);//val[i]为面额，wei[i]为重量
        }
        f[0]=0;//因为此处假设的是小猪储钱罐 恰好装满 的情况
        //注意初始化（要求恰好装满背包，那么在初始化时除了f[0]为0其它f[1..V]均设为-∞，
        //这样就可以保证最终得到的f[N]是一种恰好装满背包的最优解。
        //如果并没有要求必须把背包装满，而是只希望价格尽量大，初始化时应该将f[0..V]全部设为0）
        for(i =0 ; i < n ; i++)
        {
            for(j = wei[i] ; j <= c ; j++)
            {
                f[j] = min(f[j],f[j-wei[i]]+val[i]);//此处求的是最坏的情况所以用min，确定最少的钱,当然最后就用max了，HEHE
            }
        }
        if(f[c] == INF)
            printf("This is impossible.\n");
        else
            printf("The minimum amount of money in the piggy-bank is %d.\n",f[c]);
    }
    return 0;
}
//此代码为HDU1114;
```

2.4 MultiBag

```
//多重背包(MultiplePack)：有N种物品和一个容量为V的背包。
//第i种物品最多有n[i]件可用，每件费用是c[i]，价值是w[i]。
//求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，
//且价值总和最大。
//HDU 2191

#include <cstdio>
#include <cstring>
#define N 247
int max(int x,int y){
```

```
11      return x>y?x:y;
12  }
13  int main()  {
14      int t,n,m,i,j,k;
15      int w[N],pri[N],num[N],f[N];
16      while(~scanf("%d",&t)){
17          while(t--){
18              memset(f,0,sizeof(f));
19              scanf("%d%d",&n,&m);//n为总金额，m为大米种类
20              for(i = 0 ; i < m ; i++){
21                  scanf("%d%d%d",&pri[i],&w[i],&num[i]);//num[i]为每种大米的袋数
22              }
23              for(i = 0 ; i < m ; i++){
24                  for(k = 0 ; k < num[i] ; k++){
25                      for(j = n ; j >= pri[i]; j--){
26                          f[j] = max(f[j],f[j-pri[i]]+w[i]);
27                      }
28                  }
29              }
30              printf("%d\n",f[n]);
31          }
32      }
33      return 0;
34  }
```

3 数据结构

3.1 BTree

```
template<class T>

struct TreeNode {
    T value;
    TreeNode *left;
    TreeNode *right;
};

template<class T>
TreeNode<T> *createTree(const T *pre, const T *in, const int len) {
    TreeNode<T> *t = NULL;
    if (len > 0) {
        t = new TreeNode<T>;
        t->value = pre[0];
        int index;
        for (index = 0; index < len; index++) {
            if (in[index] == pre[0]) {
                break;
            }
        }
        if (index == len) {
            index = -1;
        }
        t->left = createTree(pre + 1, in, index);
        t->right = createTree(pre + index + 1, in + index + 1, len - index - 1);
    }
    return t;
}

template<class T>
int preOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        int count = 1;
        out.push(root->value);
        count += preOrder(root->left, out);
        count += preOrder(root->right, out);
        return count;
    } else {
        return 0;
    }
}

template<class T>
int inOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        int count = 1;
        count += inOrder(root->left, out);
        out.push(root->value);
        count += inOrder(root->right, out);
        return count;
    } else {
        return 0;
    }
}

```

```
56  template<class T>
57  void postOrder(TreeNode<T> *root, queue<T> &out) {
58      if (root) {
59          postOrder(root->left, out);
60          postOrder(root->right, out);
61          out.push(root->value);
62      } else {
63          return;
64      }
65  }
66
67  template<class T>
68  T *convertQueueToArray(queue<T> &out, int len) {
69      T *list = new T[len];
70      int now = 0;
71      while (!out.empty() && now < len) {
72          list[now] = out.front();
73          out.pop();
74          now++;
75      }
76      return list;
77  }
78
79  template<class T>
80  void destroyTree(TreeNode<T> *root) {
81      if (root) {
82          destroyTree(root->left);
83          destroyTree(root->right);
84          delete root;
85      } else return;
86  }
87
88  template<class T>
89  void insertIntoBSTree(TreeNode<T> *root, const T &value) {
90      if (!root) {
91          return;
92      }
93      if (value < root->value) {
94          if (root->left) {
95              insertIntoTree(root->left, value);
96          } else {
97              root->left = new TreeNode<T>;
98              root->left->value = value;
99              root->left->left = NULL;
100             root->left->right = NULL;
101         }
102     } else if (value > root->value) {
103         if (root->right) {
104             insertIntoTree(root->right, value);
105         } else {
106             root->right = new TreeNode<T>;
107             root->right->value = value;
108             root->right->left = NULL;
109             root->right->right = NULL;
110         }
111     }
112 }
113
114 template<class T>
```

```
TreeNode<T> *createBSTree(T *list, int len) {
    if (len < 1) {
        return NULL;
    }
    TreeNode<T> *root = new TreeNode<char>;
    root->value = list[0];
    root->left = NULL;
    root->right = NULL;
    for (int i = 1; i < len; i++) {
        insertIntoBSTree(root, list[i]);
    }
    return root;
}
```

3.2 pbds-bbtree

```
// 红黑树
tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> t;
// null_type无映射(低版本g++为null_mapped_type)
// 类似multiset
tree<int, null_type, less_equal<int>, rb_tree_tag, tree_order_statistics_node_update> t
    ;

find_by_order(size_t order);
// 结点更新
tree_order_statistics_node_update
insert(p);
erase(it);
// 求k在树中是第几大:
order_of_key(p);
// 找到第order小的迭代器
find_by_order(order);
// 前驱
lower_bound(p);
// 后驱
upper_bound(p);
// 合并
a.join(b);
// 分割 key小于等于v的元素属于a，其余的属于b
a.split(v, b);
```

4 图论

5 博弈

6 分治

7 其他