#背包问题1

```
#include <iostream>
#include <cstdio>
#include <complex>
#define A 1000010
using namespace std;
int f[A], w[A], v[A];
/*---------0-1背包----------*/
int knapsack01(int n, int V) {
    memset(f, 0xc0c0c0c0, sizeof f); f[0] = 0; //需要装满
    memset(f, 0, sizeof f); //不需要装满
    for (int i = 1; i <= n; i++)
        for (int j = V; j >= w[i]; j--)
            f[j] = max(f[j], f[j - w[i]] + v[i]);
    return f[V];
}
/*-----------完全背包----------*/
int Fullbackpack(int n, int V) {
    for (int i = 1; i <= n; i++)
        for (int j = w[i]; j <= V; j++)
            f[j] = max(f[j], f[j - w[i]] + v[i]);
    return f[V];
}
/*-------多重背包二进制拆分-------*/
int number[A];
int MultiplePack1(int n, int V) {
    for (int i = 1; i <= n; i++) {
        int num = min(number[i], V / w[i]);
        for (int k = 1; num > 0; k <<= 1) {
            if (k > num) k = num;
            num -= k;
            for (int j = V; j >= w[i] * k; j--)
            f[j] = max(f[j], f[j - w[i] * k] + v[i] * k);
        }
    }
    return f[V];
}
int newv[A], neww[A], cnt;
int MultiplePack2(int n, int V) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= c[i]; j <<= 1) {
            newv[cnt] = j * v[i];
            neww[cnt++] = j * w[i];
            c[i] -= j;
        }
        if (c[i] > 0) {
            newv[cnt] = c[i] * v[i];
            neww[cnt++] = c[i] * w[i];
        }
    }
    for (int i = 1; i <= cnt; i++)
```

```
            for (int j = V; j >= neww[i]; j--)
                f[j] = max(f[j], f[j - neww[i]] + newv[i]);
        return f[V];
}
/*------------多重背包单调队列优化------------*/
void MultiPack(int p, int w, int v) {
    for (int j = 0; j < cost; j++) {
        int head = 1,tail = 0;
        for (int k = j, i = 0; k <= V / 2; k += w, i++) {
            int r = f[k] - i * v;
            while (head <= tail and r >= q[tail].v) tail--;
            q[++tail] = node(i, r);
            while (q[head].id < i - num) head++;
            f[k] = q[head].v + i * v;
        }
    }
}
/*-----------二维费用背包----------*/
int t[A], g[A], dp[B][B];
int Costknapsack(int n, int V, int T) {
    for (int i = 1; i <= n; i++)
        for (int j = T; j >= w[i]; j--)
            for (int k = V; k >= g[i]; k--)
                dp[j][k] = max(dp[j][k], dp[j - w[i]][k - g[i]] + v[i]);
        return dp[T][V];
}
/*--------------分组背包--------------*/
int a[B][B];
int Groupingbackpack() {
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]);
    for (int i = 1; i <= n; i++)
        for (int j = m; j >= 0; j--)
            for (int k = 1; k <= j; k++)
                f[j] = max(f[j], f[j - k] + a[i][k]);
    return f[m];
}
/*------------K优解---------------*/
int kth(int n, int V, int k) {
        for (int i = 1; i <= n; i++) {
            for (int j = V; j >= w[i]; j--) {
                    for (int l = 1; l <= k; l++) {
                            a[l] = f[j][l];
                            b[l] = f[j - w[i]][l] + v[i];
                        }
                        a[k + 1] = -1;
                        b[k + 1] = -1;
                        int x = 1, y = 1, o = 1;
                        while (o != k + 1 and (a[x] != -1 or b[y] != -1)) {
                                if (a[x] > b[y]) f[j][o] = a[x], x++;
                                else f[j][o] = b[y], y++;
                                if (f[j][o] != f[j][o - 1]) o++;
                        }
                }
        }
        return f[V][k];
}
```