一、简答

1. 状态转移时，同时考虑转移是否选择该背包的信息，伪代码：

```
vector vis[v];
// 记录背包选择信息

for (i : n) {
    for (j : v) {
        if (可以放下背包 i && 放下 i 后价值更优) {
            vis[j] = vis[j - i.w];
            vis[j].push_back(i);
        }
    }
}
```

二、代码及注释

E.超大背包

题目大意

背包问题。
数据范围比较特殊：

$$N(1 \leq N \leq 40),\ V(1 \leq V \leq 10^{15})$$

$$w[i](1 \leq w[i] \leq 10^{15}),\ c[i](1 \leq c[i] \leq 10^{15})$$

解法

分治，将背包分为两组，求对应所有子集的体积占用以及价值总和，问题转化为两组中各选取一个子集，满足体积条件下价值最大。组内排序，删除次优子集。遍历第一组，二分寻找第二组中与之组合体积满足条件，且价值最大的子集，更新答案。

时间复杂度

枚举子集为 $O(2^{n/2})$,枚举二分时 $O(2^{n/2} * log_2(2^{n/2})) = O(n2^{n/2})$,因此总复杂度为 $O(n2^{n/2})$.

代码

```cpp
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
using ll = long long;

// 2^40
vector<pair<ll, ll> > s1, s2;

// 枚举子集
vector<pair<ll, ll> > cal(vector<pair<ll, ll> > k) {
    vector<pair<ll, ll> > res;

    int sz = k.size();
    if (sz < 1) return res;
    // 位运算判断是否选择
    for (int i = 0; i < (1 << sz); i++) {
        ll W = 0, C = 0;
        for (int j = 0; j < sz; j++) {
            if ((i >> j) & 1) {
                W += k[j].first;
                C += k[j].second;
            }
        }
        res.push_back({W, C});
    }
    return res;
}

// 比较函数，重量越小且相同重量，价值更大的优先级更大
bool cmp(pair<ll, ll> p, pair<ll, ll> q) {
    if (p.first != q.second)
        return p.first < q.first;
    return p.second > q.second;
}

// 排除次优子集方案
vector<pair<ll, ll> > cl(vector<pair<ll, ll> > k) {
    vector<pair<ll, ll> > res;
    int sz = k.size();
    if (sz < 1) return res;
    res.push_back({k[0].first, k[0].second});
    for (int i = 1; i < sz; i++) {
        ll C = res.back().second;
        // 由于重量为递增排序，价值也必须是严格递增，否则为次优子集
        if (k[i].second > C) {
            res.push_back({k[i].first, k[i].second});
        }
    }
    return res;
```

```
}

void solve() {
    int n, v;
    cin >> n >> v;

    ll w, c;
    // 将前20个背包分为一组，其余归为另一组
    for (int i = 1; i <= min(n, 20); i++) {
        cin >> w >> c;
        s1.push_back({w, c});
    }
    for (int i = 1; i <= n - 20; i++) {
        cin >> w >> c;
        s2.push_back({w, c});
    }

    s1 = cal(s1);
    s2 = cal(s2);

    sort(s1.begin(), s1.end(), cmp);
    sort(s2.begin(), s2.end(), cmp);

    s1 = cl(s1);
    s2 = cl(s2);

    ll ans = 0;
    // 若输入不足20个背包，可以直接输出
    if (n <= 20) {
        for (int i = 0; i < s1.size(); i++) {
            if (v >= s1[i].first) ans = max(ans, s1[i].second);
        }
        cout << ans << '\n';
        return;
    }

    int sz = s1.size();
    // 枚举第一个分组
    for (int i = 0; i < sz; i++) {
        if (s1[i].first > v) break;
        int pos = -1;
        int l = 0, r = s2.size();
        // 在第二组中寻找匹配分组（二分）
        while (l <= r) {
            int mid = (l + r) >> 1;
            // 若满足体积限制，则记录答案
            // 由于分组排序特点，答案只会是更优的，直接更新即可
            if (s2[mid].first + s1[i].first <= v) {
                pos = mid;
                l = mid + 1;
            }
```

```
            else r = mid - 1;
        }

        if (pos == -1) {
            // 没有合适匹配分组，自己作为答案进行更新
            ans = max(ans, s1[i].second);
        }
        else {
            ans = max(ans, s1[i].second + s2[pos].second);
        }
    }
    cout << ans << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
    return 0;
}
```