Algorithm & Applied Probability

Jun's Blog

Unbounded Knapsack

JULY 18, 2012 LEAVE A COMMENT (HTTPS://ALGOMATH.WORDPRESS.COM/2012/07/18/UNBOUNDED-KNAPSACK/#RESPOND)

After we solved 0-1 knapsack, we have already got some basic idea how to use DP to address knapsack problem. Let's dig deeper.

Unbounded Knapsack: We have n items. Each type of item has a value v_i. Each type of time has a cost c_i. The difference between 01 knapsack and unbounded knapsack is that there is no upper limit on each type of item.

Same as 01 knapsack, we let d(i,w) to denote the maximal value we can get from first i types of items with weight constraint w. At each time, we will decide how many items for one type to pick, while in 01 knapsack we only need to decide : pick or not pick.

$$dp(i,w) = max\{dp(i-1, w-kc_i) + kv_i | 0 \leq kc_i \leq w\}$$

Similarly, we can optimize the space by changing the above equation.

$$dp(i,w) = max(dp(i-1,w), dp(i, w-c_i) + v_i)$$

***You should pay attention to this conversion. It's very important to understand why it works.

```
int unboundedKnapsack(int c[], int v[], int n, int W){
    vector<int> dp(W+1);
    memset(dp,0,(W+1)*sizeof(int));
    int i,j;
    for(i=1;i<=n;i++){
        for(j=c[i];j<=W;j++)
            dp[j]=max(dp[j],dp[j-c[i]]+v[i]);
    }
    return dp[W];
}
```

FILED UNDER DYNAMIC PROGRAMMING