CS208 Lab9 Practice

12312110 李轩然

DDL: May.31

A Task-scheduling Problem

Desciption

A unit-time task is a job, such as a program to be run on a computer, that requires exactly one unit of time to complete. Given a finite set S of unit-time tasks, a schedule for S is a permutation of S specifying the order in which to perform these tasks. The first task in the schedule begins at time 0 and finishes at time 1, the second task begins at time 1 and finishes at time 2, and so on.

The problem of **scheduling unit-time tasks with deadlines and penalties for a single processor** has the following inputs:

A set $S = \{a_1, a_2, \ldots, a_n\}$ of n unit-time tasks;

A set of n integer deadlines $d_1, d_2, \ldots, d_n$, such that each d_i satisfies $1 \leq d_i \leq n$ and task a_i is supposed to finish by time d_i;

A set of n nonnegative weights or penalties $w_1, w_2, \ldots, w_n$, such that we incur a penalty of w_i if task a_i is not finished by time d_i, and we incur no penalty if a task finishes by its deadline.

We wish to find a schedule for S that minimizes the total penalty incurred for missed deadlines.

Analysis

It's 01 backpack. Suppose $dp[i][j]$ represents in the first i tasks, we choose j tasks, the max sum of w. So if we do not choose i-th task, $dp[i][j] = dp[i-1][j]$; if we choose,
$dp[i][j] = dp[i-1][j-1] + w[i]$.

Time complexity: $O(n^2)$.

C++ Code

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int d[n+1], w[n+1];
    for (int i = 1; i <= n; i++)
        cin >> d[i] >> w[i];

    int dp[n+1][n+1];
```

```
    for (int i = 0; i <= n; i++)
        for (int j = 0; j <= n; j++)
            dp[i][j] = 0;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 0; j <= n; j++)
        {
            dp[i][j] = dp[i-1][j];
            if (j > 0 and j <= d[i]) dp[i][j] = max(dp[i][j], dp[i-1][j-1] + w[i]);
        }
    }

    int maxx = 0, total = 0;
    for (int j = 0; j <= n; j++)
        maxx = max(maxx, dp[n][j]);

    for (int i = 1; i <= n; i++)
        total += w[i];

    cout << total - maxx;
    return 0;
}
```

Another Task-scheduling Problem

Description

The problem of **scheduling non-unit time tasks with deadlines and penalties for a single processor** has the following inputs:

A set $S = \{a_1, a_2, \ldots, a_n\}$ of n non-unit time tasks;

A set of n integer time $t_1, t_2, \ldots, t_n$, to finish a_i need t_i time;

A set of n integer deadlines $d_1, d_2, \ldots, d_n$, such that each d_i satisfies $1 \leq d_i \leq n$ and task a_i is supposed to finish by time d_i;

A set of n nonnegative weights or penalties $w_1, w_2, \ldots, w_n$, such that we incur a penalty of w_i if task a_i is not finished by time d_i, and we incur no penalty if a task finishes by its deadline.

We wish to find a schedule for S that minimizes the total penalty incurred for missed deadlines.

Analysis & C++ Code

Same, just change the value from 1 to t_i.

$$dp[i][j] = \begin{cases} dp[i-1][j], & \text{do not choose } i \\ dp[i-1][j-t[i]] + w[i], & \text{choose } i \end{cases}$$

Time complexity: $O(n \sum t_i)$.

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int d[n+1], t[n+1], w[n+1];
    int maxx = 0, total = 0, sumt = 0;
    for (int i = 1; i <= n; i++)
    {
        cin >> d[i] >> t[i] >> w[i];
        sumt += t[i];
        total += w[i];
    }

    int dp[n+1][n+1];
    for (int i = 0; i <= n; i++)
        for (int j = 0; j <= n; j++)
            dp[i][j] = 0;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 0; j <= sumt; j++)
        {
            dp[i][j] = dp[i-1][j];
            if (j > 0 and j <= d[i]) dp[i][j] = max(dp[i][j], dp[i-1][j-t[i]] + w[i]);
        }
    }

    for (int j = 0; j <= sumt; j++)
        maxx = max(maxx, dp[n][j]);

    cout << total - maxx;
    return 0;
}
```