Tareas

Autora: Elena V. R.

Usamos la técnica de programación dinámica. Sea $dp[i][j]$ igual al mínimo tiempo necesario para completar j tareas, teniendo en cuenta sólo las primeras $i-1$ tareas. Tenemos la siguiente relación de recurrencia:

$$dp[i+1][j+1] = \begin{cases} \mathbf{mín}(dp[i][j+1], dp[i][j] + t_i) & \text{si } dp[i][j] \leq p_i \\ dp[i][j+1] & \text{si } dp[i][j] > p_i \end{cases}$$

Usándola, podemos calcular la DP en tiempo $O(n^2)$ (y memoria $O(n)$ si eliminamos la primera dimensión). La respuesta es el mayor valor de j tal que $dp[n+1][j] < \infty$.

```cpp
#include<bits/stdc++.h>

using namespace std;

const int INF = 1000000001;

int main() {
    int T;
    cin >> T;
    while (T--) {
        int n;
        cin >> n;
        vector<int> t(n);
        vector<int> p(n);
        for (int i=0; i < n; ++i) cin >> t[i] >> p[i];
        vector<int> dp(n+1, INF);
        dp[0] = 0;
        for (int i=0; i < n; ++i) {
            for (int j=n-1; j > -1; --j) {
                if (dp[j] <= p[i]) {
                    dp[j+1] = min(dp[j+1], dp[j] + t[i]);
                }
            }
        }
        int ans = n;
        while (dp[ans] == INF) ans--;
        cout << ans << endl;
    }
}
```