Contest programming

Dynamic programming

Maksym Planeta

04.05.2018

Table of Contents

Outline

1. Introduction
2. Linear data structures. Long arithmetic.
3. Greedy and Divide and conquer algorithms. Dynamic programming.
4. Dynamic programming (practice).
5. Practice. Easy Tasks. Dynamic Programming. Math.
6. Mathematical and combinatorial problems.
7. Algorithms on graphs
8. Algorithms on graphs
9. Practice session
10. Contest

10130 – Supersale

Task

Every person can take as many objects (only one object of each kind), as he/she can carry out from the SuperSale. We have given list of objects with prices and their weight. We also know, what is the maximum weight that every person can stand. What is the maximal value of objects we can buy at SuperSale?

10130 – Supersale

Input

The input consists of T test cases. The number of them ($1 \leq T \leq 1000$) is given on the first line of the input file. Each test case begins with a line containing a single integer number N that indicates the number of objects ($1 \leq N \leq 1000$). Then follows N lines, each containing two integers: P and W. The first integer ($1 \leq P \leq 100$) corresponds to the price of object. The second integer ($1 \leq W \leq 30$) corresponds to the weight of object. Next line contains one integer ($1 \leq G \leq 100$) its the number of people in our group. Next G lines contains maximal weight ($1 \leq MW \leq 30$) that can stand this i-th person from our family ($1 \leq i \leq G$).

Output

Output For every test case your program has to determine one integer. Print out the maximal value of goods which we can buy with that family.

Solution – 0 – 1 Knapsack

State

(Item index, Weight)

Formula

$$Value(i, w) = \max(Value(i+1, w), Value(i+1, w - weight_i) + value_i)$$

Code – Value Function

```
int p(int w, int idx) {
  if (w == 0) return 0;
  if (idx == items.size()) return 0;

  int cur_w = items[idx].first;
  int cur_p = items[idx].second;

  if (memo[idx * 31 + w] > 0)
      return memo[idx * 31 + w];

  if (cur_w > w) return p(w, idx + 1);

  int res = max(p(w, idx + 1),
                cur_p + p(w - cur_w, idx + 1));

  return memo[idx * 31 + w] = res;
}
```

GCD

Euclidean algorithm

$$\gcd(a, b) \begin{cases} a & b = 0 \\ \gcd(b, a \bmod b) & b \neq 0 \end{cases}$$

Extended Euclidean Algorithm

Goal

$$d = ax + by = \gcd(a, b)$$

Extended-Euclidean(a, b)

```
1  if b == 0
2        return (a, 1, 0)
3  (d', x', y') = Extended-Euclidean(b, a mod b)
4  (d, x, y) = (d', y', x' − ⌊a/b⌋y')
5  return (d, x, y)
```

Extended Euclidean Algorithm

a	b	$\lfloor a/b \rfloor$	d	x	y
99	78	1	3	-11	14
78	21	3	3	3	-11
21	15	1	3	-2	3
15	6	2	3	1	-2
6	3	2	3	0	1
3	0		3	1	0

Prime numbers

Primality tests:

- Trivial: $O(\sqrt{N}/\log N)$
- Miller-Rabin: $O(s)$
- Sieve of Eratosthenes: $O(N \log\log N)$ (for N numbers)

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21

2 3 5 7 9 11 13 15 17 19 21

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21

2 3 5 7 9 11 13 15 17 19 21

2 3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21

2 3 5 7 9 11 13 15 17 19 21

2 3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~

2 3 5 7 11 13 17 19

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21

2 3 5 7 9 11 13 15 17 19 21

2 3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~

2 3 5 7 11 13 17 19

2 3 5 7 11 13 17 19

Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21

2 3 5 7 9 11 13 15 17 19 21

2 3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~

2 3 5 7 11 13 17 19

2 3 5 7 11 13 17 19

2 3 5 7 11 13 17 19

Practice

Solve following set of problems in a group:

1. 12577 – Hajj-e-Akbar
2. 10114 – Loansome Car Buyer
3. 00573 – Save Setu
4. 00183 – Bit Maps
5. 11827 – Maximum GCD
6. 11368 – Nested Dolls
7. 10759 – Dice throwing
8. 10140 – Prime Distance

Home reading

Cormen.

1. Recommended
   - Section 15. Dynamic Programming

Literature