Problem Solving Techniques 문제해결

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

In-Class Problem: Shoemaker's Problem

- Can you make an efficient solution for the shoemaker's problem?

Problem

- Shoemaker's problem
  - A shoemaker has N orders from customers which he must satisfy. The shoemaker can work on only one job in each day, and jobs usually take several days. For the ith job, the integer Ti ($1 \le Ti \le 1,000$) denotes the number of days it takes the shoemaker to finish the job.
  - But popularity has its price. For each day of delay before starting to work on the ith job, the shoemaker has agreed to pay a fine of Si ($1 \le Si \le 10,000$) cents per day. Help the shoemaker by writing a program to find the sequence of jobs with minimum total fine.

Problem

- Input
  - The input begins with a single positive integer on a line by itself indicating the number of the test cases, followed by a blank line. There is also a blank line between two consecutive cases.
  - The first line of each case contains an integer reporting the number of jobs N, where $1 \leq N \leq 9$. The ith subsequent line contains the completion time Ti and daily penalty Si for the ith job.
- Output
  - For each test case, your program should print the sequence of jobs with minimal fine. Each job should be represented by its position in the input. All integers should be placed on only one output line and each pair separated by one space. If multiple solutions are possible, print the first one in lexicographic order.

Problem

- Sample input

1

4

3 4

1 1000

2 2

5 5

- Sample output

2 1 3 4

Why?

2 1 3 4
⇒ 1*4 + 4*2 + 6*5 = 42

1 2 3 4
=> 3*1000 + 4*2 + 6*5 = 3038

4 3 2 1
⇒ 5*2 + 7*1000 + 8*4 = 7042

2 1 4 3
=> 1*4 + 4*5 + 9*2 = 42
(also yields the minimum)

Problem

- Let's solve the shoemaker's problem using backtracking. This means, you should investigate all possible sequences of jobs, in order to find the sequence of jobs with minimum total fine.

- How?

  3 4

  1 1000

  2 2

  5 5

```
bool finished = FALSE;              /* found all solutions yet? */

backtrack(int a[], int k, data input)
{
        int c[MAXCANDIDATES];       /* candidates for next position */
        int ncandidates;            /* next position candidate count */
        int i;                      /* counter */

        if (is_a_solution(a,k,input))
                process_solution(a,k,input);
        else {
                k = k+1;
                construct_candidates(a,k,input,c,&ncandidates);
                for (i=0; i<ncandidates; i++) {
                        a[k] = c[i];
                        backtrack(a,k,input);
                        if (finished) return;   /* terminate early */
                }
        }
}
```

Revisit: Shoemaker's Problem

- Input
  - The input begins with a single positive integer on a line by itself indicating the number of the test cases, followed by a blank line. There is also a blank line between two consecutive cases.
  - The first line of each case contains an integer reporting the number of jobs N, where $1 \leq N \leq 1000$. The ith subsequent line contains the completion time Ti and daily penalty Si for the ith job.

$1 \leq N \leq 9$ in the previous problem

- Output
  - For each test case, your program should print the sequence of jobs with minimal fine. Each job should be represented by its position in the input. All integers should be placed on only one output line and each pair separated by one space. If multiple solutions are possible, print the first one in lexicographic order.

Revisit: Shoemaker's Problem

- This problem cannot be solved using backtracking because N can be as large as 1000, which yields 1000! cases to be investigated.

- This necessitates an efficient solution for this problem.

Revisit: Shoemaker's Problem

- Let's develop an efficient solution!

Revisit: Shoemaker's Problem

A 3 4

B 1 1000

C 2 2

D 5 5

- Suppose that the job sequence is {A, B, C, D} for the example.
  - The sequence yields the cost of
    - 0*4 + (0+3)*1000 + (0+3+1)*2 + (0+3+1+2)*5
- What if we change the order of two jobs, which are consecutively placed in the sequence?

Revisit: Shoemaker's Problem

A 3 4

B 1 1000

C 2 2

D 5 5

- Suppose that the job sequence is {A, B, C, D} for the example.
  - The sequence yields the cost of
    - 0*4 + (0+3)*1000 + (0+3+1)*2 + (0+3+1+2)*5
- What if we change the order of two jobs, which are consecutively placed in the sequence?
- Consider the job sequence {B, A, C, D}.
  - The sequence yields the cost of
    - 0*1000 + (0+1)*4 + (0+1+3)*2 + (0+1+3+2)*5
- Consider the job sequence {A, C, B, D}.
  - The sequence yields the cost of
    - 0*4 + (0+3)*2 + (0+3+2)*1000 + (0+3+2+1)*5

Revisit: Shoemaker's Problem

- Can you observe a property?

Revisit: Shoemaker's Problem

A 3 4

B 1 1000

C 2 2

D 5 5

- Can you observe a property?
- Suppose that the job sequence is {A, B, C, D} for the example.
  - The sequence yields the cost of
    - 0*4 + (0+3)*1000 + (0+3+1)*2 + (0+3+1+2)*5
- Consider the job sequence {B, A, C, D}.
  - The sequence yields the cost of
    - 0*1000 + (0+1)*4 + (0+1+3)*2 + (0+1+3+2)*5

Revisit: Shoemaker's Problem

A 3 4

B 1 1000

C 2 2

D 5 5

- Can you observe a property?

- Suppose that the job sequence is {A, B, C, D} for the example.
  - The sequence yields the cost of
    - 0*4 + (0+3)*1000 + (0+3+1)*2 + (0+3+1+2)*5

- Consider the job sequence {A, C, B, D}.
  - The sequence yields the cost of
    - 0*4 + (0+3)*2 + (0+3+2)*1000 + (0+3+2+1)*5

Revisit: Shoemaker's Problem

- First, we observe that
  - Although we change the order of two jobs, which are consecutively placed in the sequence, the cost for the remaining jobs after the two jobs in the sequence do not change.

Revisit: Shoemaker's Problem

- Can you observe another property?

Revisit: Shoemaker's Problem

A 3 4

B 1 1000

C 2 2

D 5 5

- Can you observe another property?
- Suppose that the job sequence is {A, B, C, D} for the example.
  - The sequence yields the cost of
    - $\boxed{0*4 + (0+3)*1000}$ + (0+3+1)*2 + (0+3+1+2)*5
- Consider the job sequence {B, A, C, D}.
  - The sequence yields the cost of
    - $\boxed{0*1000 + (0+1)*4}$ + (0+1+3)*2 + (0+1+3+2)*5

$\boxed{0*4 + (0+3)*1000}$

$\boxed{0*1000 + (0+1)*4}$

3*1000
Day[A]*Fine[B]

1*4
Day[B]*Fine[A]

Revisit: Shoemaker's Problem

A 3 4

B 1 1000

C 2 2

D 5 5

- Can you observe another property?
- Suppose that the job sequence is {A, B, C, D} for the example.
  - The sequence yields the cost of
    - 0*4 + $\boxed{(0+3)*1000 + (0+3+1)*2}$ + (0+3+1+2)*5
- Consider the job sequence {A, C, B, D}.
  - The sequence yields the cost of
    - 0*4 + $\boxed{(0+3)*2 + (0+3+2)*1000}$ + (0+3+2+1)*5

$\boxed{(0+3)*1000 + (0+3+1)*2}$

$\boxed{(0+3)*2 + (0+3+2)*1000}$

1*2
Day[B]*Fine[C]

2*1000
Day[C]*Fine[B]

Revisit: Shoemaker's Problem

- Second, we observe that
  - If we change the order of two jobs (a and b), which are consecutively placed in the sequence, the cost difference is Day[a]*Fine[b] - Day[b]*Fine[a].
    - We can judge which order is better between {a, b} and {b, a}.

Revisit: Shoemaker's Problem

- How can we use the following two properties to solve the problem efficiently?

- First, we observe that
  - Although we change the order of two jobs, which are consecutively placed in the sequence, the cost for the remaining jobs after the two jobs in the sequence do not change.

- Second, we observe that
  - If we change the order of two jobs (a and b), which are consecutively placed in the sequence, the cost difference is Day[a]*Fine[b] - Day[b]*Fine[a].
    - We can judge which order is better between {a, b} and {b, a}.

Revisit: Shoemaker's Problem

- How can we use the following two properties to solve the problem efficiently?

- Solution
  - P1. Start from any job sequence.
  - P2. Find any two consecutive jobs that reduce the total fine by comparing Day[a]*Fine[b] - Day[b]*Fine[a].
  - P3. repeat P2 until there is no such job pair.

Revisit: Shoemaker's Problem

- How can we use the following two properties to solve the problem efficiently?
- Solution
  - P1. Start from any job sequence.
  - P2. Find any two consecutive jobs that reduce the total fine by comparing Day[a]*Fine[b] - Day[b]*Fine[a].
  - P3. repeat P2 until there is no such job pair.
- Any better, systematic way?

Revisit: Shoemaker's Problem

- The previous solution is equivalent to sort jobs with Day[a]*Fine[b] - Day[b]*Fine[a], and choose the jobs in the sorted order.

  - Bubble sort!
    - ABCD
    - Day[A]*Fine[B] – Day[B]*Fine[A]=3*1000-1*4>0
    - BACD
    - Day[A]*Fine[C] – Day[C]*Fine[A]=3*2-2*4<0
    - BACD
    - Day[C]*Fine[D] – Day[D]*Fine[C]=2*5-5*2=0

A 3 4

B 1 1000

C 2 2

D 5 5

Revisit: Shoemaker's Problem

- The above solution is equivalent to sort jobs with Day[a]*Fine[b] - Day[b]*Fine[a], and choose the jobs in the sorted order.

A 3 4

B 1 1000

C 2 2

D 5 5

  - Bubble sort!
    - ABCD
    - Day[A]*Fine[B] – Day[B]*Fine[A]=3*1000-1*4>0
    - BACD
    - Day[A]*Fine[C] – Day[C]*Fine[A]=3*2-2*4<0
    - BACD
    - Day[C]*Fine[D] – Day[D]*Fine[C]=2*5-5*2=0
    - BACD
    - BACD
    - BACD

  - Therefore, the final solution is
    - BACD

Revisit: Shoemaker's Problem

- How can we address the following constraints?
  - If multiple solutions are possible, print the first one in lexicographic order.

- What is time-complexity?

Revisit: Shoemaker's Problem

- How can we address the following constraints?
  - If multiple solutions are possible, print the first one in lexicographic order.
  - Answer: Apply a proper tie breaking rule.
- What is time-complexity?
  - Answer: the same as sort jobs