JOIG 春合宿 2-1
メロンの出荷

解説：熊田順一

問題概要

メロンが N 個あり、i 番目のメロンの重さは A_i

x 番目のメロンから N 番目のメロンまで、以下のように箱詰めする。

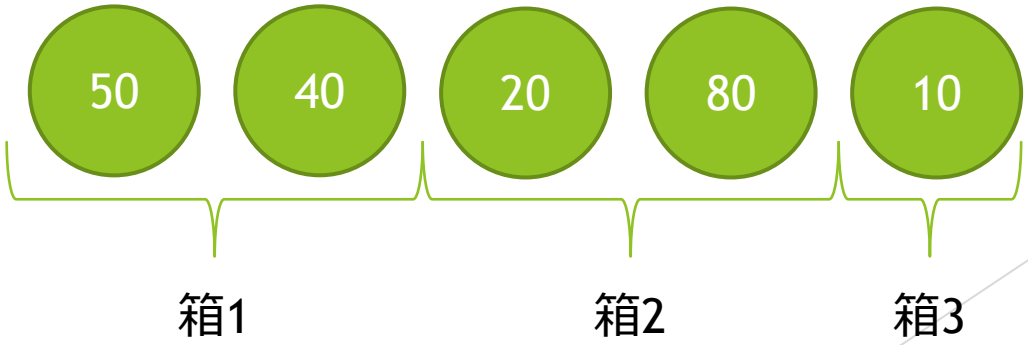
- 箱のメロンの重さの合計が L 以下の間は箱に詰める
- 箱に詰めたら L を超える場合は、そのメロンは箱に詰めないで、箱を出荷する。
  - 詰めなかったメロンは次の箱に詰める。
- N番目のメロンを箱に詰めたら、その箱を出荷する。

x = 1, 2, …, N それぞれについて、出荷した箱の個数と、最後に出荷した箱に入っているメロンの総重量を求めよ。

入出力例

入力例	出力例
7 100	4 10
20	4 10
80	3 10
50	2 90
40	2 10
20	1 90
80	1 10
10	

ex) x = 3 の時



小課題 1 ($A_1 = A_2 = \cdots = A_N$, 6点)

$A_1 = A_2 = \ldots = A_N = A$ とする。

また、[a] で a 以下の最大の整数を表す。(例：[0.1] = 0, [2.4] = 2)

箱に詰められるメロンの個数は [L/A] で一定。

この値を M とすると、出荷する箱の個数は (N-x+1) / M 以上の最小の整数 = [(N - x + M) / M] 。

また、最後に詰めるメロンの個数は、

- (N - x + 1) が M の倍数の時 M 個
- そうでない時、個数は (N - x + 1) を M で割ったあまり

各 x = 1,2,..., N について、定数時間で答えを計算できるため、全体の計算量は $O(N)$

小課題1 回答例

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int N, L;
    scanf("%d %d", &N, &L);
    int A;
    scanf("%d", &A); // 2つ目以降の値はいらない
    int M = L / A;
    for (int x = 1; x <= N; x++) {
        printf("%d ", (N - x) / M + 1);
        if ((N - x + 1) % M == 0) {
            printf("%d¥n", M * A);
        } else {
            printf("%d¥n", ((N - x + 1) % M) * A);
        }
    }
}
```

小課題2 ($N \leq 1000$, 21点)

各 x について、題の通りにシミュレートする。

- 出荷した箱の個数を num とし、作業中の箱にあるメロンの重さの和を now とする。
- 始め、num = 0, now = 0 としておく。
- 重さ a のメロンを詰める時、
  - now + a <= L であれば now に a を加える。(そのまま詰める。)
  - そうでない時、 num を 1 増やし、 now の値を a に更新する。(今の箱を出荷し、新しい箱に詰める。)
- N 番目までのメロンまで繰り返した後の、num + 1, now が答え。

計算量は $O(N^2)$

小課題2 回答例

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int N, L;
    scanf("%d %d", &N, &L);
    vector<int> A(N);
    for (auto& a : A) {
        scanf("%d", &a);
    }
    for (int i = 0; i < N; i++) {
        int num = 0, now = 0;
        for (int j = i; j < N; j++) {
            if (now + A[j] <= L) now += A[j];
            else {
                num++;
                now = A[j];
            }
        }
        printf("%d %d¥n", num + 1, now);
    }
}
```

小課題3 (出荷する箱の数は 10 個以下, 29点)

i 番目のメロンから始めたら、j 番目のメロンを入れる直前で箱を出荷する

⇔ j は $A_i + A_{i+1} + \ldots + A_j > L$ を満たす最小の i 以上の整数

⇔ j は $(A_1 + A_2 + \ldots + A_j) - (A_1 + A_2 + \ldots + A_{i-1}) > L$ を満たす最小の整数

よって、$sum_i = A_1 + A_2 + \ldots + A_{i-1}$ ($i = 0$ の時 $sum_i = 0$) とすると、

1. $sum_{j+1} > sum_i + L$ を満たす最小の j を二分探索で探す。
    1. なければ、N まで箱に詰めて終了。
2. あれば、j - 1 番目のメロンまで箱に詰めて、j 番目のメロンから同様に調べていく。

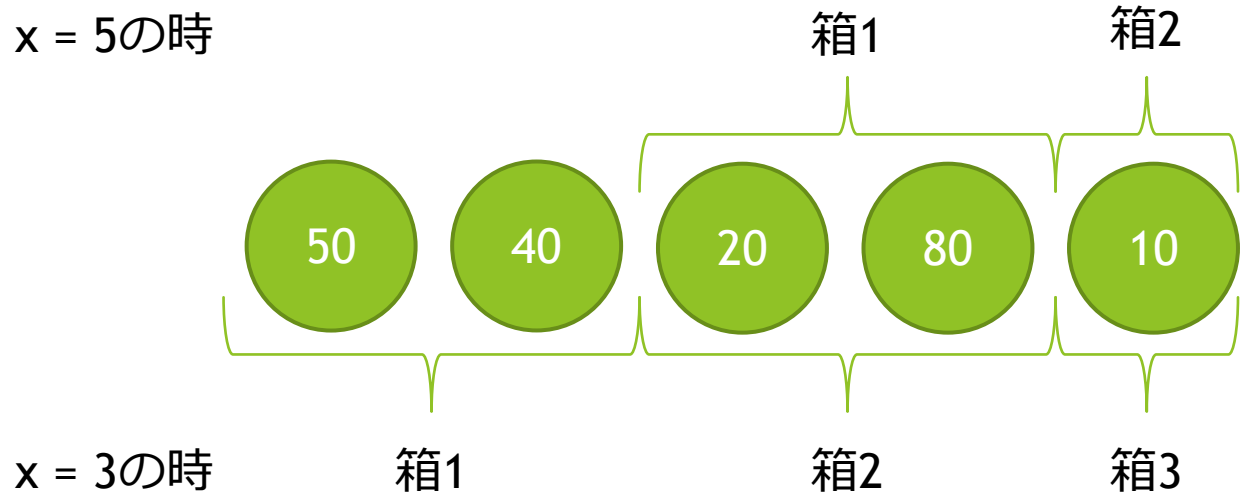
とすることで、$O(NM\log N)$ で計算できる。

ただし M は出荷する箱の個数。

実装する際は、オーバーフローに注意。(sum の値は32bit 整数に収まらない場合がある。)

小課題4 導入

何か気づくことはありませんか？



小課題4 (箱に詰めるメロンは最大10個、33点)

メロンを詰め切った次のメロンから始めた場合と状況がよく似ている→ DP で解ける。

- dp1[i] := x=i とした時の、出荷した箱の個数
- dp2[i] := x=i とした時の、最後に出荷した箱に入っていたメロンの総重量

とすると、

i 番目のメロンから N 番目のメロンまで一つの箱に詰められる時、

- dp1[i] = 1, dp2[i] = $A_i + A_{i+1} + \ldots + A_N$

i 番目のメロンから始めて、次の箱に最初に詰めるのが j 番目のメロンの時、

- dp1[i] = dp1[j] + 1, dp2[i] = dp2[j]

よって、i の値を N→1 と移動させ、最初の箱だけシミュレーションすることで、$O(NC)$ で解ける。

ただし C は一つの箱に詰めたメロンの個数の最大値。

小課題5 ($N \leq 200000$, 11点)

小課題3 の解法と 小課題4 の解法を組み合わせる。

小課題4 では次の箱に最初に入れるメロンのインデックスを愚直に求めていた。
これを小課題3 の二分探索に置き換えることで、高速に計算する。

後は小課題4 でやった通りにすることで、時間内に計算することができる。

計算量は $O(N\log N)$

小課題3 と同様、オーバーフローに注意。

回答例

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
int main() {
    int N;
    ll L;
    scanf("%d %lld", &N, &L);
    vector<ll> A(N), sum(N + 2), dp1(N + 1), dp2(N + 1);
    for (int i = 0; i < N; i++) scanf("%lld", &A[i]);
    sum[0] = 0;
    for (int i = 0; i < N; i++) sum[i + 1] = sum[i] + A[i];
    sum[N + 1] = (ll)1e18;
    dp1[N] = 0;
    dp2[N] = -1;
    for (int i = N - 1; i >= 0; i--) {
        // sum[fl] > sum[i] + L を満たす最小の fl を計算
        int fl = upper_bound(sum.begin(), sum.end(), sum[i] + L) - sum.begin();
        dp1[i] = dp1[fl - 1] + 1;
        // 出荷する箱が一つかどうかで場合分け
        dp2[i] = (dp1[i] > 1 ? dp2[fl - 1] : sum[fl - 1] - sum[i]);
    }
    for (int i = 0; i < N; i++) printf("%lld %lld¥n", dp1[i], dp2[i]);
}
```

小課題5 の別解

小課題4 で愚直に求めていた j は、尺取法で全体で O(N) で計算できる

- ひとつ前で求めた j から、合計が L 以下になるまでシミュレーションをする。

全体として計算量 O(N) で計算することも可能

回答例

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int N, L;
    scanf("%d %d", &N, &L);
    vector<int> A(N), dp1(N + 1), dp2(N + 1);
    for (int i = 0; i < N; i++) scanf("%d", &A[i]);
    int j = N;
    int sum = 0;
    dp1[N] = dp2[N] = 0;
    for (int i = N - 1; i >= 0; i--) {
        sum += A[i];
        while (sum > L) {
            j--;
            sum -= A[j];
        }
        dp1[i] = dp1[j] + 1;
        dp2[i] = (j == N ? sum : dp2[j]);
    }
    for (int i = 0; i < N; i++) printf("%d %d¥n", dp1[i], dp2[i]);
}
```

尺取法

得点分布 (スライドだけ)