ABC 115 解説

出題・解説: @evima0

2018 年 12 月 8 日

A: Christmas Eve Eve Eve

正解までの道のりを細かく分割すると次の通りです。

1. 標準入力から整数 D を受け取る。
2. X から何らかの方法で答えとなる文字列 `Christmas...` を得る。
3. 得た文字列を標準出力に出す。

手順 1, 3 については practice contest 問題 A の言語別サンプルコードが参考になります。手順 2 については if 文を使う、for 文を $25 - D$ 周回す (100 点問題の要求レベルを少し超える)、文字列に + や * といった演算子を使う (あれば) といった手が考えられます。C++, Python3 による実装例を示します。

```cpp
#include <iostream>
using namespace std;
int main(){
    int D;
    cin >> D;
    string ans;
    if(D == 25) ans = "Christmas";
    else if(D == 24) ans = "Christmas Eve";
    else if(D == 23) ans = "Christmas Eve Eve";
    else ans = "Christmas Eve Eve Eve";
    cout << ans << endl;
}
```

```python
print('Christmas' + ' Eve' * (25 - int(input())))
```

B: Christmas Eve Eve

「N 個の商品の定価の合計」を S 円、「N 個の商品のうち最も定価が高いものの定価」を M 円とすると答えは $S - M/2$ で、この S と M を実際に求めてみよという問題です。

この問題が A: Christmas Eve Eve Eve より難しいのは、商品の個数が 2 個や 3 個などと固定されておらず、N 個 $(2 \leq N \leq 10)$ と可変であるためです。これに対応すべく、何らかの「ループを生む機構」を使うことになります。代表例が for 文で、この内部で足し算や大小比較を行うことで S や M が求められます。

S を求めるためのより具体的な手順を述べると、まず S という値を宣言して 0 で初期化します。そして $i = 1, i = 2, \ldots, i = N$ のそれぞれに対して S に p_i を足すと、最終的に S に求めたい値が入ります。M についても同様で、最初に M という値を 0 で初期化し[*1]、「M と p_i を比較して p_i の方が大きければ M を p_i に置き換える」ことを繰り返せば求まります。これを C++ で実装したコードが以下です。

```cpp
#include <algorithm>    // max
#include <iostream>
using namespace std;
int main(){
    int N;
    cin >> N;
    int S = 0, M = 0;
    for(int i = 0; i < N; ++i){  // ここでは商品番号を 0, 1, ..., N-1 とする
        int p;
        cin >> p;
        S += p;
        M = max(M, p);  // M と p のうち大きい方。かなり便利！ ちなみに小さい方は min
    }
    int ans = S - M / 2;
    cout << ans << endl;
}
```

言語によっては、純粋な for 文以外の機構も便利です。以下のコード (Python3) をご覧ください。

```python
N = int(input())
p = [int(input()) for i in range(N)]
print(sum(p) - max(p) // 2)
```

[*1] すべての p_i より小さい値なら 0 でなくても可

C: Christmas Eve

木の本数は最大で 10 万本とかなり多く、K も 2 以上 $N-1$ 以下の範囲で自由な値を取りうるため、例えば $N = 100000, K = 50000$ のとき飾る木の選び方の総パターン数は天文学的な数値になります[*2]。したがって木の飾り方の全パターンを 2 秒で調べることはできず、選択肢の数を減らす必要があります。

やや唐突ですが、仮に N 本の木が低い順に左から一列に並んでいたらどうでしょう？「飾った最も高い木」(=飾った最も左の木) と「飾った最も低い木」(=飾った最も右の木) の高さをできるだけ近づけるには、ずばり K 本の隣り合った木を飾るべきです。もしそうせずに飾った木と飾った木の間に飾っていない木があるなら、「間を詰める」ことにより状況が改善されるか、少なくとも悪くはならない[*3]からです。

よって、仮に N 本の木が低い順に左から一列に並んでいた場合、「左から $1, 2, \ldots, K$ 本目を飾る」「$2, 3, \ldots, K+1$ 本目を飾る」...「$N-K+1, N-K+2, \ldots, N$ 本目を飾る」という $N-K+1$ 通りのパターンをすべて試せば十分です。それぞれのパターンにおける $h_{\max} - h_{\min}$ の値も、最も高い木は最も右、最も低い木は最も左にあるため簡単に求まります。

残る問題は、実際には N 本の木が低い順に入力されるとは限らないことです。これは与えられた木の高さを低い順に並び替えてしまえば (ソートすれば) 解決します。ほとんどの主要なプログラミング言語は標準機能として高速に動作する[*4]ソートを備えているので、これを利用しましょう。[*5]

```
N, K = map(int, input().split())
h = [int(input()) for i in range(N)]
h.sort()
print(min(h[i+K-1] - h[i] for i in range(N-K+1)))
```

[*2] 252060836892200 . . . 通り (30101 桁)
[*3] 同じ高さの木が複数並んでいる場合
[*4] 長さ n の列に対し $O(n \log_2 n)$ 時間で動作する
[*5] ソートを「自前」で適当に実装してしまうと、$N = 100000$ のとき 2 秒以内に実行が終わりません。高速に動作するソートの実装は AtCoder Beginner Contest の範囲を超えた課題で、興味があれば Wikipedia などが参考になるでしょう。

D: Christmas

「レベル i バーガーの厚さ (層の総数)」を a_i、「レベル i バーガーに含まれるパティの総数」を p_i とします $(i \geq 0)$。これらの値は $a_0 = 1, a_i = 2a_{i-1} + 3$ $(i \geq 1)$, $p_0 = 1, p_i = 2p_{i-1} + 1$ $(i \geq 1)$ と順次求められます。

求めるべき数、「レベル N バーガーの下から X 層に含まれるパティの枚数」を $f(N, X)$ と書きます。$N = 0$ の場合は問題ないので、$N \geq 1$ の場合を考えましょう。$N = 0$ の場合が問題ないことを活用して、「レベル $N-1$ バーガーについてなら何を聞かれてもわかる。つまり、どんな整数 Y を指定されても $f(N-1, Y)$ を知れる」と仮定してみると、次のようにレベル N バーガーについても何を聞かれてもわかってしまいます。

1. $X = 1$ のとき: $f(N, X) = 0$ (一番下のバンだけ)
2. $1 < X \leq 1 + a_{i-1}$ のとき: $f(N, X) = f(N-1, X-1)$
   (一番下のバン + その上のレベル $N-1$ バーガーの下から $X-1$ 層)
3. $X = 2 + a_{i-1}$ のとき: $f(N, X) = p_{i-1} + 1$
   (一番下のバン + その上のレベル $N-1$ バーガー + その上のパティ)
4. $2 + a_{i-1} < X \leq 2 + 2a_{i-1}$ のとき: $f(N, X) = p_{i-1} + 1 + f(N-1, X-2-a_{i-1})$
   ((前略) + その上のパティ + その上のレベル $N-1$ バーガーの下から $X-2-a_{i-1}$ 層)
5. $X = 3 + 2a_{i-1}$ のとき: $f(N, X) = 2p_{i-1} + 1$
   ((前略) + その上のパティ + その上のレベル $N-1$ バーガー + その上のバン)

これをそのまま再帰関数として実装すれば、$N+1$ 回以下の計算ステップで答えが求まります。

```python
N, X = map(int, input().split())

a, p = [1], [1]
for i in range(N):
    a.append(a[i] * 2 + 3)
    p.append(p[i] * 2 + 1)

def f(N, X):  # X <= 0 や X > a_N を許容し解説本文から簡略化
    if N == 0:
        return 0 if X <= 0 else 1
    elif X <= 1 + a[N-1]:
        return f(N-1, X-1)
    else:
        return p[N-1] + 1 + f(N-1, X-2-a[N-1])

print(f(N, X))
```