展覧会 2 [Exhibition 2]

日本情報オリンピック女性部門（JOIG 2021）

2021.4.25

解説：E869120

1 問題概要

1

いくつかの絵が廊下に置かれている

各絵には価値が付けられている

2

N 個の絵のうち M 個を残し

その他をすべて取り外す

絵の間隔を全部 D 以上にせねばならない

1 問題概要

1

いくつかの絵が廊下に置かれている

各絵には価値が付けられている

2

N 個の絵のうち M 個を残し

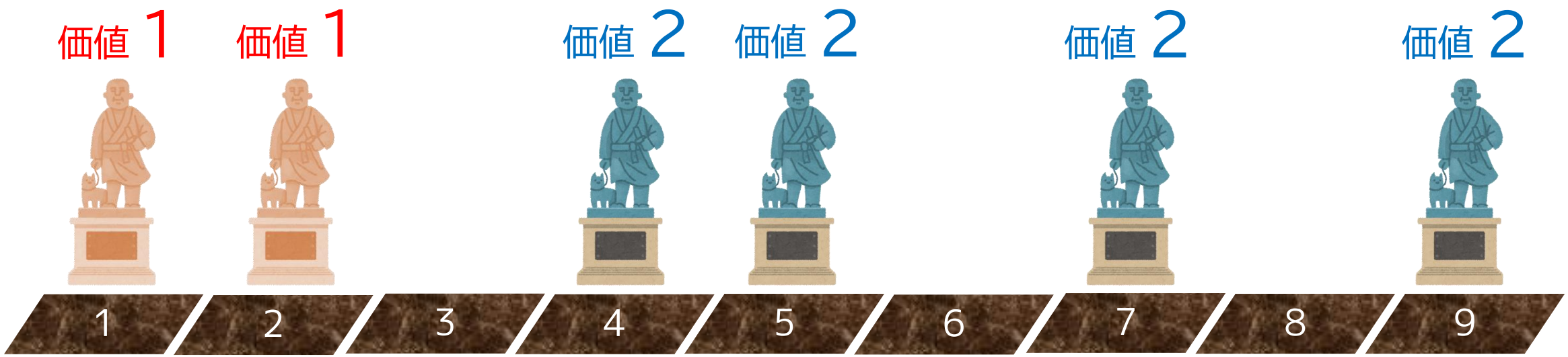
その他をすべて取り外す

価値が最小となる絵の価値を

最大化したい

絵の間隔を全部 K 以上にせねばならない

1 具体例



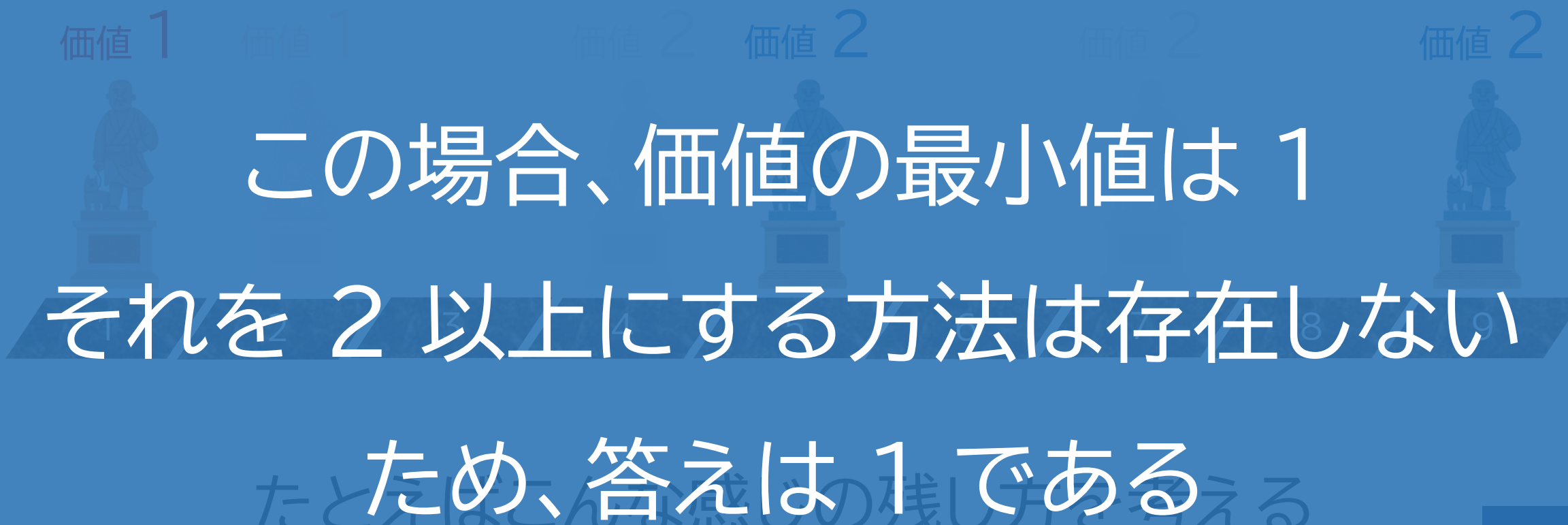
$M = 3$ 個の絵を残すが

距離は $D = 4$ 以上離す必要がある

1 具体例

たとえばこんな感じの残し方を考える

1 具体例



この場合、価値の最小値は 1

それを 2 以上にする方法は存在しない

ため、答えは 1 である

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	
2	$N = M$	
3	$N \leq 15$	
4	$N \leq 1000, V_i = 2$	
5	$N \leq 1000$	
6	$N \leq 10^5$	

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	
3	$N \leq 15$	
4	$N \leq 1000, V_i = 2$	
5	$N \leq 1000$	
6	$N \leq 10^5$	

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	絵を座標の小さい順にソートしてチェックする
3	$N \leq 15$	
4	$N \leq 1000, V_i = 2$	
5	$N \leq 1000$	
6	$N \leq 10^5$	

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	絵を座標の小さい順にソートしてチェックする
3	$N \leq 15$	bit 全探索をしよう
4	$N \leq 1000, V_i = 2$	
5	$N \leq 1000$	
6	$N \leq 10^5$	

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	絵を座標の小さい順にソートしてチェックする
3	$N \leq 15$	bit 全探索をしよう
4	$N \leq 1000, V_i = 2$	「答えは2か？」→「答えは1か？」
5	$N \leq 1000$	
6	$N \leq 10^5$	

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	絵を座標の小さい順にソートしてチェックする
3	$N \leq 15$	bit 全探索をしよう
4	$N \leq 1000, V_i = 2$	「答えは2か？」→「答えは1か？」
5	$N \leq 1000$	答えは $V_1, V_2, \ldots, V_N$ のいずれかなので全探索する
6	$N \leq 10^5$	

1 小課題ごとの解法概要

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	絵を座標の小さい順にソートしてチェックする
3	$N \leq 15$	bit 全探索をしよう
4	$N \leq 1000, V_i = 2$	「答えは2か？」→「答えは1か？」
5	$N \leq 1000$	答えは $V_1, V_2, \ldots, V_N$ のいずれかなので全探索する
6	$N \leq 10^5$	答えで二分探索

1 小課題ごとの解法概要

キーワードだけでは分からないと思うので

以降のスライドでは小課題ごとに詳しく解説します

目次

2 小課題 1 | $M = 1$

1 個しか絵を残せないので、価値が最も大きい絵を残すのが最適

2 小課題 1 | $M = 1$

1 個しか絵を残せないので、価値が最も大きい絵を残すのが最適

答えは $\max(V_1, V_2, \ldots, V_N)$ となり、実装例は次の通り

実装例（C++）

```cpp
#include <iostream>
using namespace std;
int N, M, D, X[200009], V[200009];

int main() {
    cin >> N >> M >> D;
    for (int i = 1; i <= N; i++) cin >> X[i] >> V[i];
    for (int i = 1; i <= N; i++) Answer = max(Answer, V[i]);
    cout << Answer << endl;
    return 0;
}
```

3点

2 小課題 2 | $N = M$

- すべての絵を残せるので、この残し方が条件を満たすか、つまり距離が D 未満となっている絵の組が存在するか判定すれば良い

2 小課題 2 | $N = M$

- すべての絵を残せるので、この残し方が条件を満たすか、つまり距離が D 未満となっている絵の組が存在するか判定すれば良い

全部距離 D 以上の場合 → 答えは $\min(V_1, V_2, \ldots, V_N)$

2 小課題 2｜$N = M$

- すべての絵を残せるので、この残し方が条件を満たすか、つまり距離が D 未満となっている絵の組が存在するか判定すれば良い

全部距離 D 以上の場合 → 答えは $\min(V_1, V_2, \ldots, V_N)$

そうでない場合 → 答えは -1

2 小課題 2 | $N = M$

- すべての絵を残せるので、この残し方が条件を満たすか、つまり距離が K 未満となっている絵の組が存在するか判定すれば良い

どうやって判定するか？

2 小課題 2｜$N = M$（低速な解法）

- 最も単純な方法として、以下のように全部の絵の組に対して距離を計算するアルゴリズムが考えられる

実装例（C++）

```
bool check() {
    for (int i = 1; i <= N; i++) {
        for (int j = i + 1; j <= N; j++) {
            int dist = abs(X[i] - X[j]);
            if (dist < D) return false;
        }
    }
    return true;
}
```

2 小課題 2｜$N = M$（低速な解法）

- 最も単純な方法として、以下のように全部の絵の組に対して距離を計算するアルゴリズムが考えられる

実装例（C++）

```
bool check() {
    for (int i = 1; i <= N; i++) {
        for (int j = i + 1; j <= N; j++) {
            int dist = abs(X[i] - X[j]);
            if (dist < D) return false;
        }
    }
    return true;
}
```

しかしこれだと TLE する

2 小課題 2｜$N = M$（低速な解法）

- 一般に、競技プログラミングにおいて、計算回数は $10^8 \sim 10^9$ 回程度までしか許されない

2 小課題 2｜$N = M$（低速な解法）

- 一般に、競技プログラミングにおいて、計算回数は $10^8 \sim 10^9$ 回程度までしか許されない

  $N = 10^5$ で計算量 $O(N^2)$ の二重ループを書くと、計算回数が 10^{10} 程度となり、数秒以内に実行が終わらず TLE となってしまう

2 小課題 2｜$N = M$（低速な解法）

- 一般に、競技プログラミングにおいて、計算回数は 10^8 ～ 10^9 回程度までしか許されない
  - N = 10万で計算量 $O(N^2)$ のアルゴリズムを書くと、計算回数が 10^{10} 程度となり、数秒以内に実行が終わらず TLE となってしまう

TLE をどう回避するか？

2 小課題 2｜$N = M$（高速な解法）

- そこで、あらかじめ、絵を座標の小さい順に並び替えておくことを考える

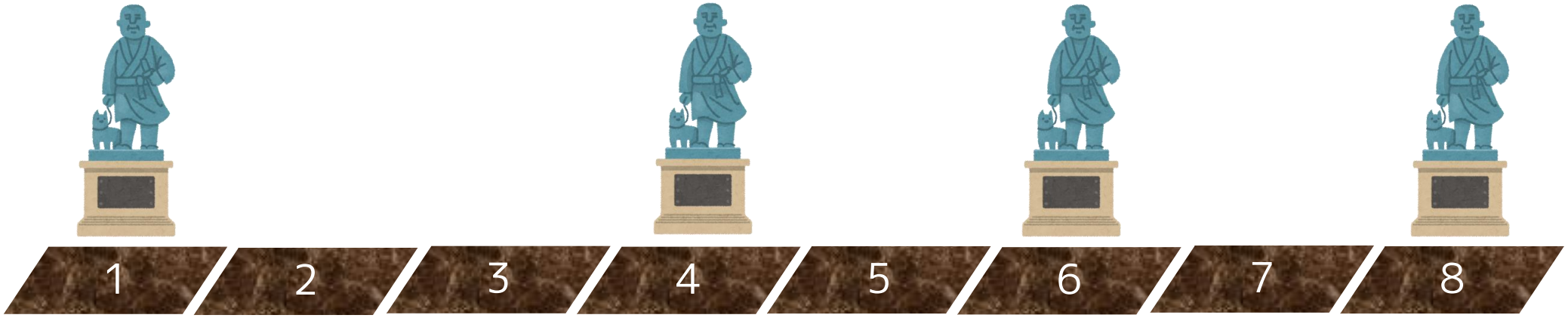
2 小課題 2｜$N = M$（高速な解法）

- そこで、あらかじめ、絵を座標の小さい順に並び替えておくことを考える
  - C++ では std::sort 関数を使うと実装できる
  - 計算量は $O(N \log N)$ です、知らない人は是非調べてみましょう

2 小課題 2｜$N = M$（高速な解法）

- 最も距離が短いのは隣り合う 2 つの絵であるはずなので、$i = 1, 2, \ldots, N - 1$ について「左から i 番目と $i + 1$ 番目の距離が K 以上かどうか」をチェックすれば良い

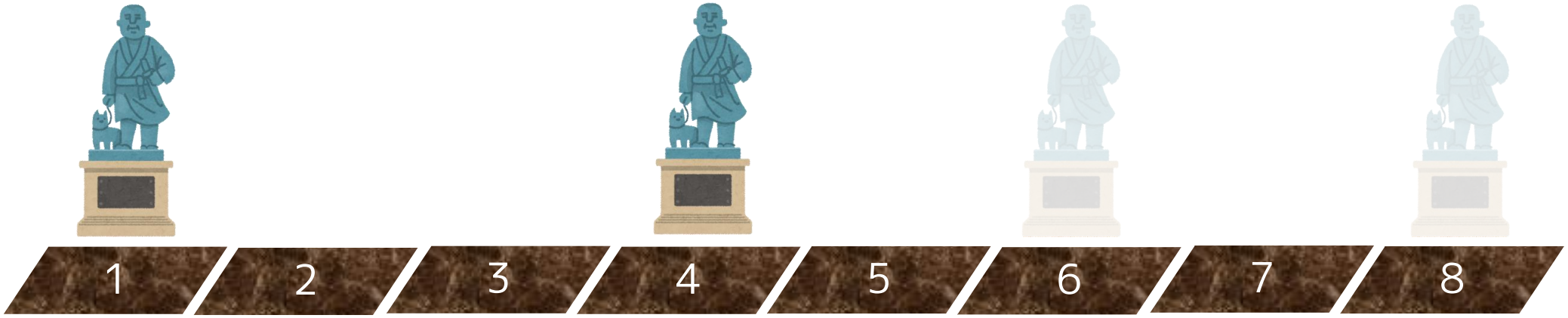
2 小課題 2｜具体例



このような場合を考える

絵間距離の最小値はいくつか？

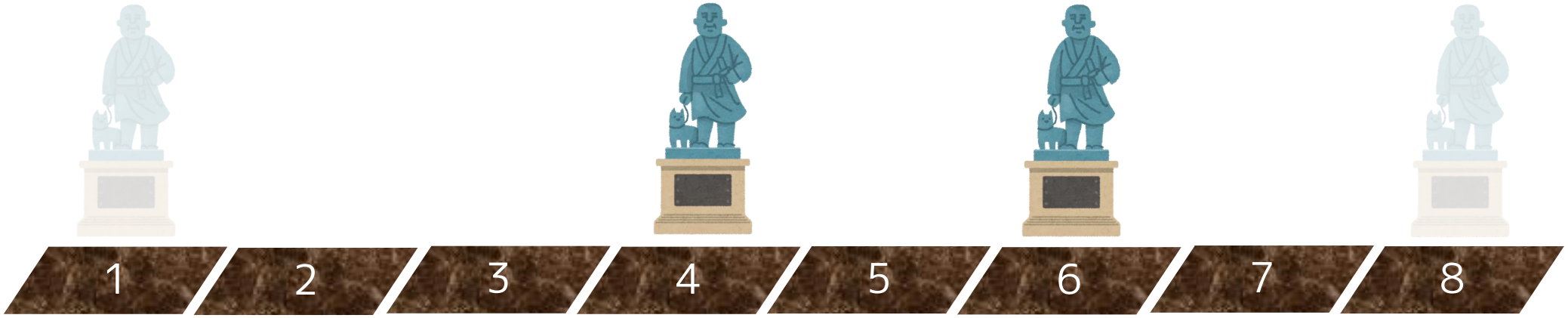
2 小課題 2｜具体例



$i = 1$ の場合：絵 1 と絵 2 の距離は $|4 - 1| = 3$

現在の距離の最小値 ＝ 3

2 小課題 2｜具体例



$i = 2$ の場合：絵 2 と絵 3 の距離は $|6 - 4| = 2$

現在の距離の最小値 ＝ 2

2 小課題 2｜具体例

$i = 3$ の場合：絵 3 と絵 4 の距離は $|8 - 6| = 2$

現在の距離の最小値 ＝ 2

2 小課題 2｜具体例

こんな感じで距離の最小が 2 と分かる

これが K 未満であれば答えは「-1」

$i = 3$ の場合：絵 3 と絵 4 の距離は $|6 - 6| = 2$

現在の距離の最小値 ＝ 2

2 小課題 2｜実装例

たとえば次のような実装が考えられる

15点

実装例（C++）

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, D, Answer = (1 << 30);
int X[200009], V[200009];

bool check() {
    for (int i = 1; i <= N - 1; i++) {
        if (X[i + 1] - X[i] < D) return false;
    }
    return true;
}

int main() {
    cin >> N >> M >> D;
    for (int i = 1; i <= N; i++) {
        cin >> X[i] >> V[i];
        Answer = max(Answer, V[i]);
    }
    sort(X + 1, X + N + 1);

    if (check() == true) cout << Answer << endl;
    else cout << "-1" << endl;
    return 0;
}
```

目次

3 小課題 3

- 制約は $N \leq 15$ とかなり小さい

3 小課題 3

- 制約は $N \leq 15$ とかなり小さい
  - 絵の残し方を全通り調べ上げる「全探索アルゴリズム」が通用するのではないか？

3 小課題 3

- 制約は $N \leq 15$ とかなり小さい

  絵の残し方を全通り調べ上げる「全探索アルゴリズム」が通用するのではないか？

  絵の残し方は 2^N 通り存在するため、$N = 15$ では高々 $32{,}768$ 通り。判定パートの計算量は小課題 2 で説明した通り $O(N \log N)$ なので、余裕を持って実行時間に間に合う。

3 小課題 3｜全探索の実装方法

- 全探索の実装方法として、例えば次のようなものが考えられる

実装例（C++）

```
bool check() {
    if (N == 15) {
        for (int i1 = 0; i1 <= 1; i1++) {
            for (int i2 = 0; i2 <= 1; i2++) {
                for (int i3 = 0; i3 <= 1; i3++) {
                    for (int i4 = 0; i4 <= 1; i4++) {
                        :
                    }
                }
            }
        }
    }
}
```

t 個目の絵を選ぶ場合は $i_t = 1$

選ばない場合は $i_t = 0$

N の値で場合分けして N 重ループを書く

3 小課題 3｜全探索の実装方法

- 全探索の実装方法として、例えば次のようなものが考えられる

実装例（C++）

```
bool check(
    if (N =
        for (int i1 = 0; i1 <= 1; i1++) {
            for (int i2 = 0; i2 <= 1; i2++) {
                for (int i3 = 0; i3 <= 1; i3++) {
                    for (int i4 = 0; i4 <= 1; i4++) {
                        :
                    }
                }
            }
        }
    }
}
```

実装が大変になってしまう！

どうしよう？？？

3 小課題 3｜より良い実装

- 「bit 全探索」という手法を使う

3 小課題 3｜より良い実装

- 「bit 全探索」という手法を使う
  - $i = 0, 1, 2, 3, \ldots, 2^N - 1$ とループを回して i を 2 進数で表したときに
  - 下から j 桁目が 1 である場合 j 個目の絵を選び、そうでなければ選ばない

3 小課題 3｜より良い実装

- 「bit 全探索」という手法を使う

  $i = 0, 1, 2, 3, \ldots, 2^N - 1$ とループを回して i を 2 進数で表したときに

  下から j 桁目が 1 である場合 j 個目の絵を選び、そうでなければ選ばない

i の値	0	1	2	3	4	5	6	7
2 進数の値	000	001	010	011	100	101	110	111
選ぶ絵	{}	{1}	{2}	{1,2}	{3}	{1,3}	{2,3}	{1,2,3}

3 小課題 3｜より良い実装

- 整数 i を 2 進数で表したときの下から $j+1$ 桁目の値は、
  ビット演算を用いて次のように計算できる

3 小課題 3｜より良い実装

- 整数 i を 2 進数で表したときの下から $j+1$ 桁目の値は、
  ビット演算を用いて次のように計算できる
  - $(i \text{ and } 2^j) = 0$、つまり「`(i & (1 << j)) == 0`」のとき、$j+1$ 桁目は 0
  - $(i \text{ and } 2^j) \neq 0$、つまり「`(i & (1 << j)) != 0`」のとき、$j+1$ 桁目は 1

3 小課題 3｜より良い実装

- 例えば以下のようなコードで 2 進数表記の j 桁目の値がわかる

実装例（C++）

```
void binary_henkan(int x) {
    int bit[15];
    for (int i = 0; i < 15; i++) {
        if ((x & (1 << i)) == 0) bit[i] = 0;
        if ((x & (1 << i)) != 0) bit[i] = 1;
    }
}
```

bit[i] は 2 進数での $i+1$ 桁目

例えば $x = 9$ の場合

bit = [1, 0, 0, 1, 0, 0, 0, …]

3 小課題 3｜解法まとめ

- まとめると、2^N 通りの方法を全探索することで、計算量 $O(2^N \times N^2)$ または $O(2^N \times N \log N)$ で解ける
  - 判定パートは小課題 2 の通り、$O(N^2)$ または $O(N \log N)$ かかる
  - 実装のひとつの方法として「bit 全探索」というものが挙げられる

3 小課題 3｜実装例

たとえば次のような実装が考えられる

34点

実装例（C++）

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, D, Answer = -1;
int X[200009], V[200009];

int check(vector<int> x) {
    // {x[0], x[1], …} の選び方が OK かの判定
    // OK の場合価値の最小を返す（詳細の実装は略）
}

int main() {
    cin >> N >> M >> D;
    for (int i = 0; i < N; i++) {
        cin >> X[i] >> V[i];
    }

    for (int i = 0; i < (1 << N); i++) {
        vector<int> vec;
        for (int j = 0; j < N; j++) {
            if ((i & (1 << j)) != 0) {
                vec.push_back(j);
            }
        }
        Answer = max(Answer, check(vec));
    }
    cout << Answer << endl;
    return 0;
}
```

目次

4 小課題 4｜$V_i \leq 2$ の場合

- $V_i \leq 2$ の場合、答えは「2」「1」「存在しない」のいずれか

  したがって、次のような問題が解ければよい

4 小課題 4｜$V_i \leq 2$ の場合

- $V_i \leq 2$ の場合、答えは「2」「1」「存在しない」のいずれか

  したがって、次のような問題が解ければよい

  - 価値 2 以上の絵のみを使って M 個以上の絵を残せるか？
  - 価値 1 以上の絵のみを使って M 個以上の絵を残せるか？
  - どれにも当てはまらなければ、答えは「存在しない（-1）」

4 小課題 4｜$V_i \leq 2$ の場合

- $V_i \leq 2$ の場合、答えは「2」「1」「存在しない」のいずれか
- したがって、次のような問題が解ければよい
  - 価値 2 以上の絵のみを使って M 個以上の絵を残せるか？
  - 価値 1 以上の絵のみを使って M 個以上の絵を残せるか？
  - どれにも当てはまらなければ、答えは「存在しない（−1）」

ではどうやって M 個以上残せるか判定すればよいのか？

4 小課題 4｜$V_i \leq 2$ の場合

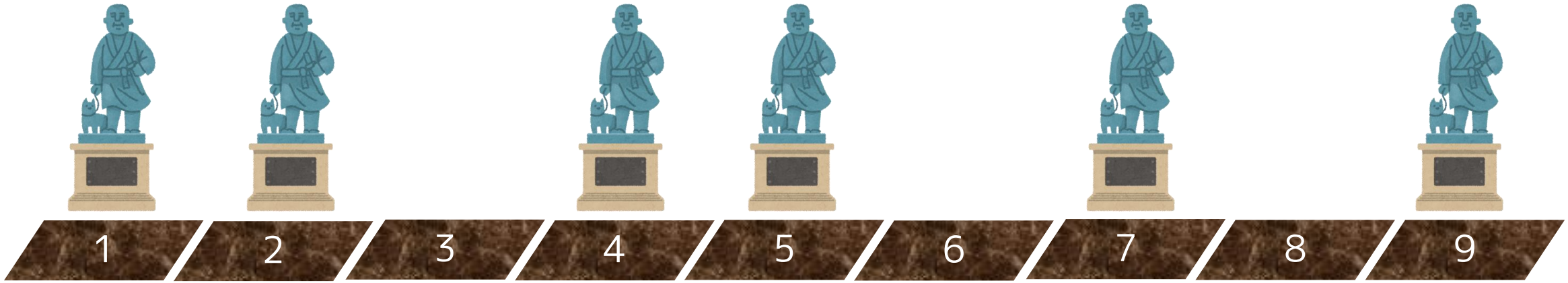
- 左にある「価値が B 以上のもの」から順に、貪欲に取っていけば良い

4 小課題 4｜$V_i \leq 2$ の場合

- 左にある「価値が B（1 または 2）以上のもの」から順に、貪欲に取っていけば良い
  - 直前に取ったものから距離が D 未満である場合、この絵を取らない
  - 直前に取ったものから距離が D 以上である場合、この絵を取る

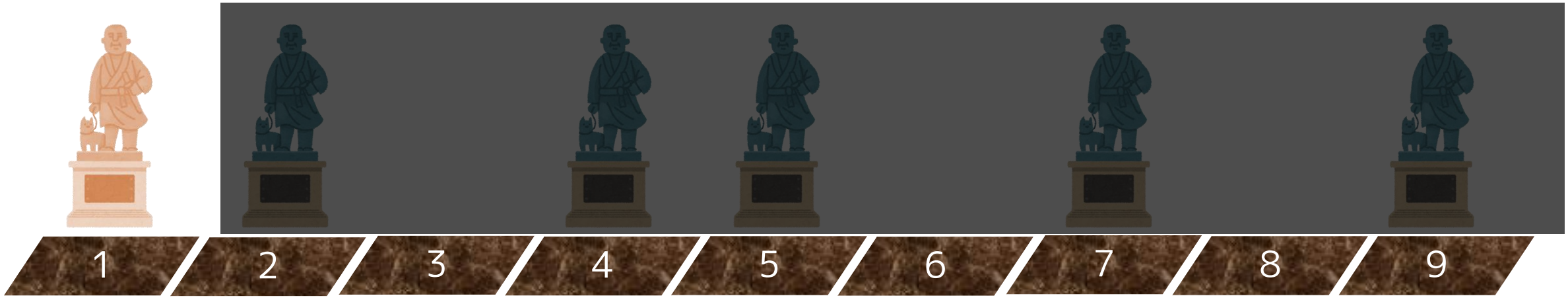
4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）



4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）

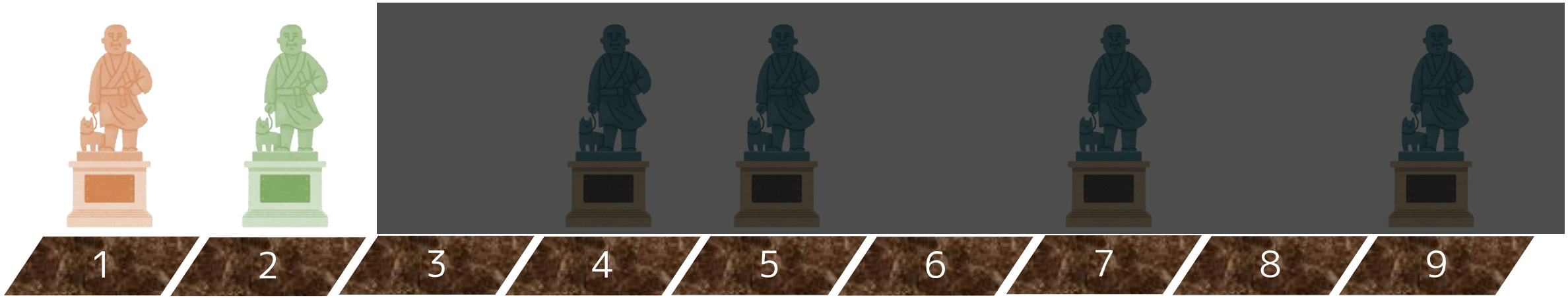


直前に残した絵は存在しない

→ この絵は残せる

4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）

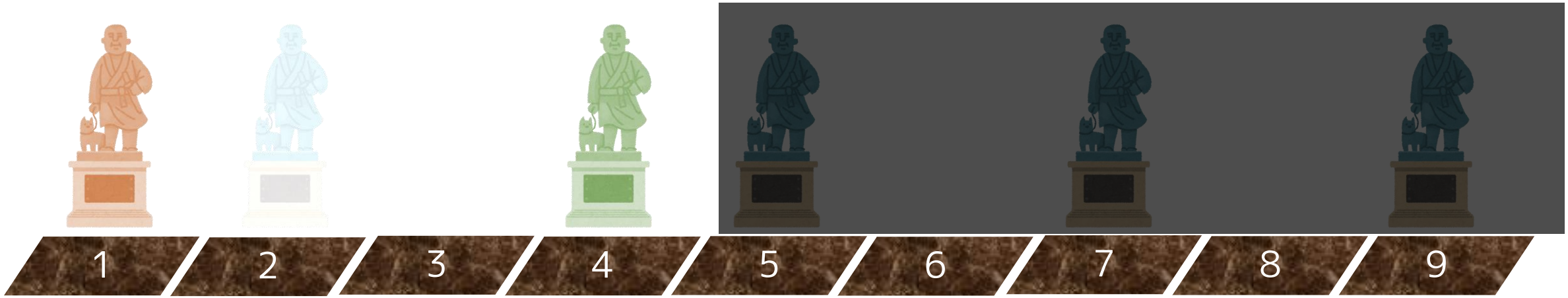


直前に残した絵と距離 1 しか離れていない

→ この絵は残せない

4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）

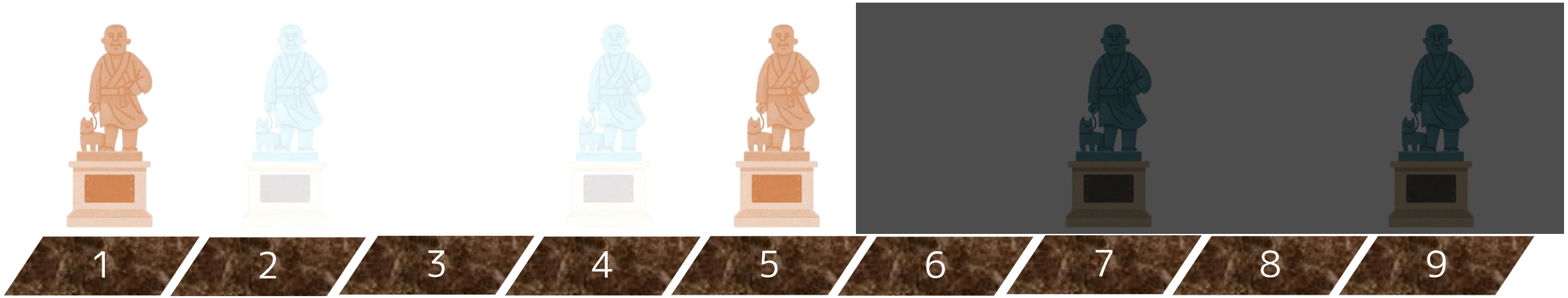


直前に残した絵と距離 3 しか離れていない

→ この絵は残せない

4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）



直前に残した絵と距離 4 離れている

→ この絵は残せる

4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）

直前に残した絵と距離 2 しか離れていない

→この絵は残せない

4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）

直前に残した絵と距離 4 離れている

→この絵は残せる

4 小課題 4｜具体例

- $M = 3$ 個以上の「価値 1 以上の絵」を取れるか？（$D = 4$ とする）

合計 3 つの絵を残せる

こんな感じで貪欲に選べばよい！

直前に残した絵と距離 4 離れている

→この絵は残せる

4 小課題 4｜$V_i \leq 2$ の場合

- これを価値の下限 $B = 2, B = 1$ について調べればよい

実装例は次の通り（$X_1 < X_2 < \cdots < X_N$ の場合）

51点

実装例（C++）

```
#include <bits/stdc++.h>
using namespace std;
int N, M, D, X[200009], V[200009];

bool check(int B) {
    int pre = -1000000000, cnt = 0;
    for (int i = 1; i <= N; i++) {
        if (V[i] < B || X[i] - pre < D) continue;
        cnt += 1; pre = X[i];
    }
    if (cnt < M) return false;
    return true;
}
int main() {
    // 入力は省略
    if (check(2) == true) printf("2¥n");
    else if (check(1) == true) printf("1¥n");
    else printf("-1¥n");
    return 0;
}
```

4 小課題 5｜$N \leq 1000$ の場合

- 実は、答え（価値の最小）は $V_1, V_2, \ldots, V_N$ のうちいずれかである

4 小課題 5｜$N \leq 1000$ の場合

- 実は、答え（価値の最小）は $V_1, V_2, \ldots, V_N$ のうちいずれかである
  - 価値が V_1 以上のものだけで M 個以上選べないか？
  - 価値が V_2 以上のものだけで M 個以上選べないか？
  - 価値が V_N 以上のものだけで M 個以上選べないか？
  - 選べたものの中で価値の下限が最も大きかったものが答え

4 小課題 5｜$N \leq 1000$ の場合

- N 回 check 関数を呼び出すので、全体計算量は $O(N^2)$

実装例（C++）

```
#include <bits/stdc++.h>
using namespace std;
int N, M, D, X[200009], V[200009], Answer = -1;

// check 関数は省略
int main() {
    // 入力は省略
    for (int i = 1; i <= N; i++) {
        if (check(V[i]) == true) Answer = max(Answer, V[i]);
    }
    cout << Answer << endl;
    return 0;
}
```

73点

4 満点解法

- 小課題 5 と同様に N 回 check 関数を呼び出すと、計算量が $O(N^2)$ となり TLE してしまう

どうしよう？？？

4 満点解法

- 小課題 5 と同様に N 回 check 関数を呼び出すと、計算量が $O(N^2)$ となり TLE してしまう

最小値の最大化は

どうしよう？？？

答えで二分探索

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

現在の答えの範囲	質問	関数呼び出し	結果例
1～16	答えは 9 以上か？	check(9)	Yes

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

現在の答えの範囲	質問	関数呼び出し	結果例
1～16	答えは 9 以上か？	check(9)	Yes
9～16	答えは 13 以上か？	check(13)	No

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

現在の答えの範囲	質問	関数呼び出し	結果例
1～16	答えは 9 以上か？	check(9)	Yes
9～16	答えは 13 以上か？	check(13)	No
9～12	答えは 11 以上か？	check(11)	No

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

現在の答えの範囲	質問	関数呼び出し	結果例
1～16	答えは 9 以上か？	check(9)	Yes
9～16	答えは 13 以上か？	check(13)	No
9～12	答えは 11 以上か？	check(11)	No
9～10	答えは 10 以上か？	check(10)	Yes

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

現在の答えの範囲	質問	関数呼び出し	結果例
1～16	答えは 9 以上か？	check(9)	Yes
9～16	答えは 13 以上か？	check(13)	No
9～12	答えは 11 以上か？	check(11)	No
9～10	答えは 10 以上か？	check(10)	Yes
10	-	-	-

4 満点解法｜答えで二分探索

- 最初、答えが 1 以上 16 以下であることが分かっていたとする

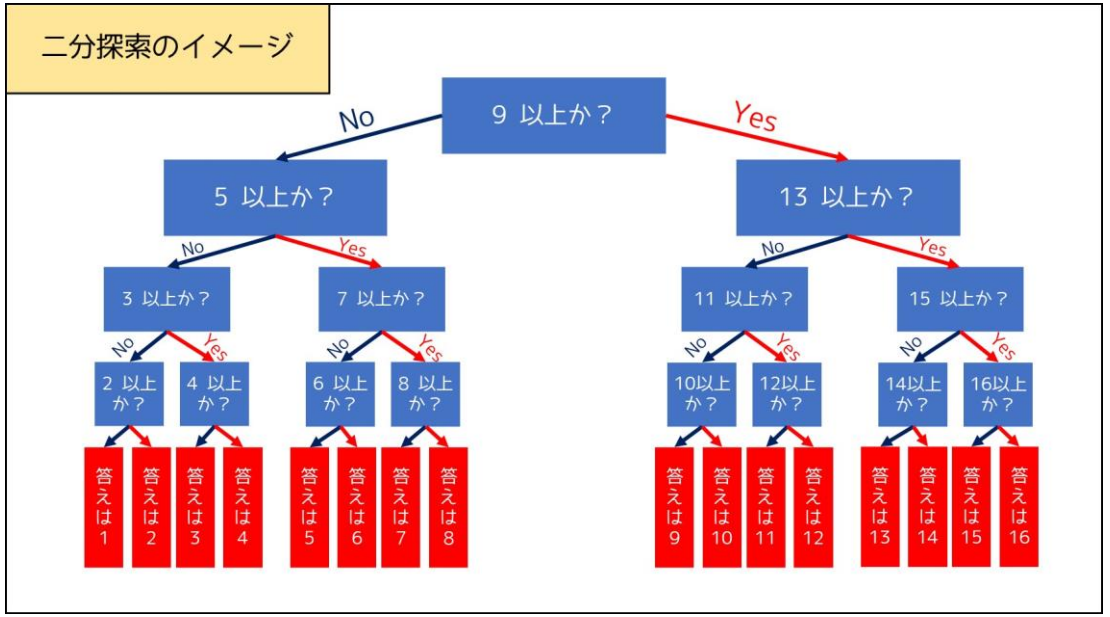
現在の答えの範囲	質問	関数呼び出し	結果例
1～16	答えは 9 以上か？	check(9)	Yes
9～16	答えは 13 以上か？	check(13)	No
9～12	答えは 11 以上か？	check(11)	No
9～10	答えは 10 以上か？	check(10)	Yes
10	-	-	-

答えの範囲を半分に絞ることで

4 回の関数呼出で答えが分かる！

4 満点解法｜答えで二分探索

- 全部まとめて図で表すとこんな感じ



https://qiita.com/e869120/items/b4a0493aac567c6a7240 より引用

4 満点解法｜答えで二分探索

- a 回 check 関数を呼び出すと、探索範囲が 2^a 分の 1 に絞られる

  答えの最大値を L として、探索範囲が L 分の 1 以下になれば答えが一通りに定まるので、

$$2^a \geq L \ (\Leftrightarrow) \ a \geq \log_2 L$$

つまり、$O(\log_2 L)$ 回 check 関数を呼び出せばよい

4 満点解法｜答えで二分探索

- 全体としては、答えで二分探索をすることで現実的な時間で実行が終わる。計算量は $O\left(N\ (\log N + \log L)\right)$ となる。

100点

目次

5 まとめ

- 各小課題において、次のような解法で解くことができる

小課題	制約	キーワード
1	$M = 1$	$\max(V_1, V_2, \ldots, V_N)$ が答え
2	$N = M$	絵を座標の小さい順にソートしてチェックする
3	$N \leq 15$	bit 全探索をしよう
4	$N \leq 1000, V_i = 2$	「答えは2か？」→「答えは1か？」
5	$N \leq 1000$	答えは $V_1, V_2, \ldots, V_N$ のいずれかなので全探索する
6	$N \leq 10^5$	答えで二分探索

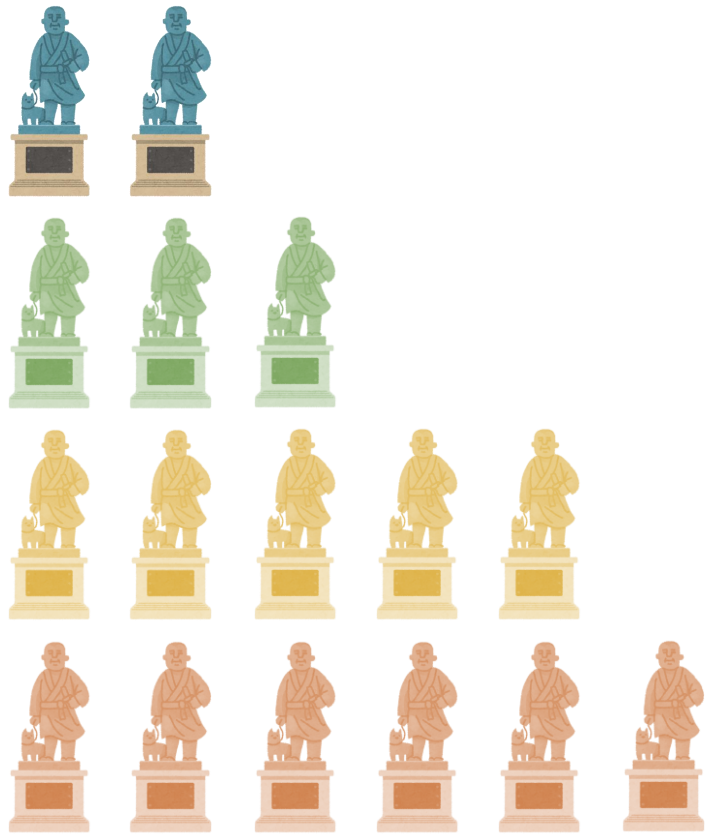
5 得点分布

100点

50～99点

20～49点

1～19点




ご清聴ありがとうございました