AtCoder Heuristic First-step Vol.2
～ビームサーチ～

600株式会社　青木栄太 (thunder)

01 問題　手法を説明するための問題例を説明

02 解法　ビームサーチを単純に適用する流れを説明

03 演習1　手を動かし、ビームサーチを使う

04 演習2　工夫し、ビームサーチを改善する

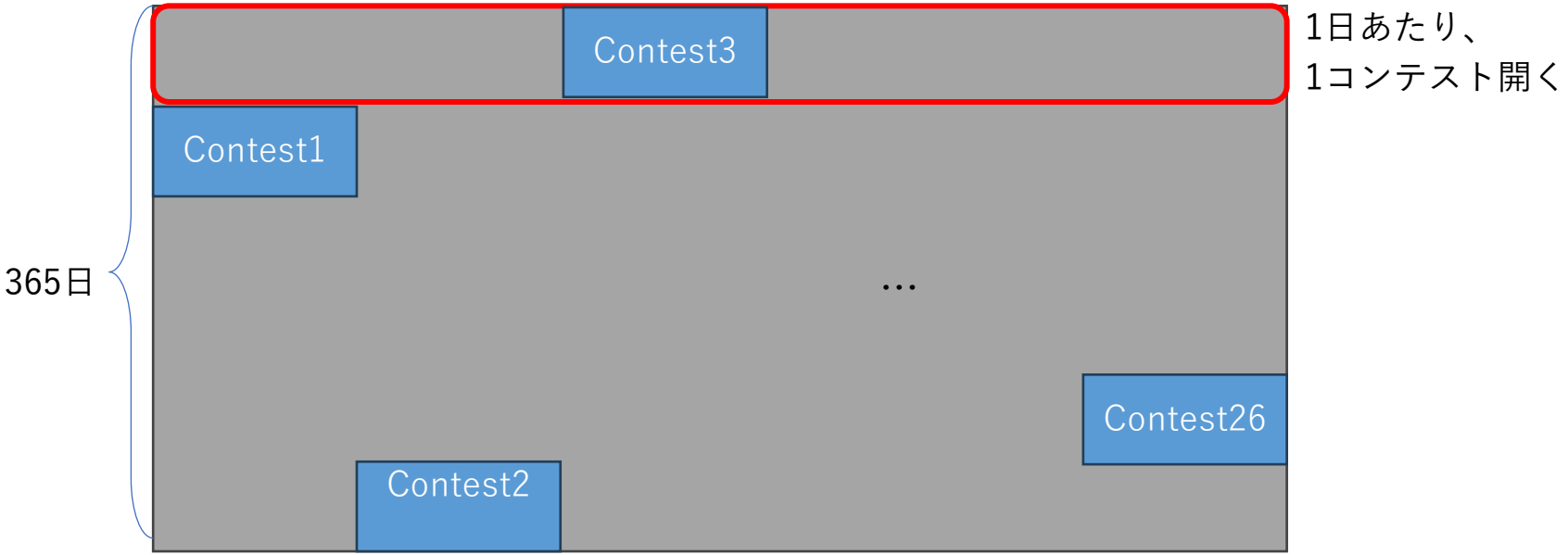
05 活用　ビームサーチを今後活用するための考え方を説明

問題

手法を説明するための
問題例を説明

問題：AtCoder Contest Scheduling　1/3

目的：365日間、26種類のコンテストから1日あたり1個ずつ開催し、
ユーザーの満足度をなるべく上げる



問題：AtCoder Contest Scheduling　2/3

d日目にコンテストiを開催すると、
あらかじめ決められた相性に応じて満足度が上がる

	Contest1	Contest2	Contest3	…	Contest26
Day1	9558	16014	17351	…	16814
Day2	9821	9867	3208	…	18702
…	…	…	…	…	…
Day365	14952	1836	6298	…	6547

2日目にContest3を開催すると、
満足度が3208上がる

問題：AtCoder Contest Scheduling　3/3

コンテストiの満足度の下がりやすさに応じて、
最後にコンテストが開催されてからの日数×下がりやすさ だけ下がる

満足度の下がりやすさ i
10

Day0	**Contest iを開催**
Day1	1
Day2	2
Day3	**Contest iを開催**
Day4	1
Day5	2
Day6	3

5日目時点、
Contest iは、3日目に開催されてから2日間開催されていない。
よって、2×10＝20だけ満足度が下がる。

01 問題

02 解法

03 演習1

04 演習2

05 活用

解法

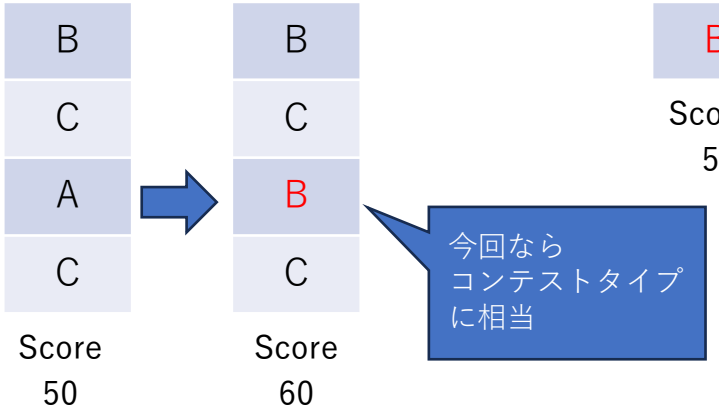
ビームサーチを

単純に適用する流れを説明

局所探索型解法と構築型解法

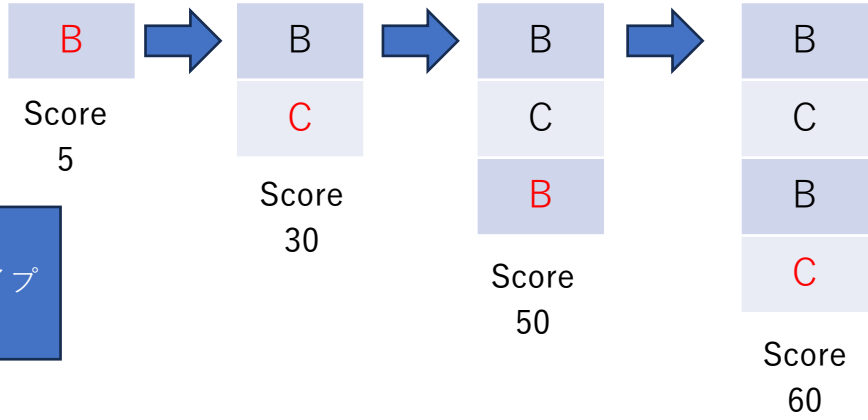
よく使う探索手法には大きく2パターンあるが、今回は構築型を解説

局所探索型
（山登り、焼きなまし）



解の一部を変えて評価

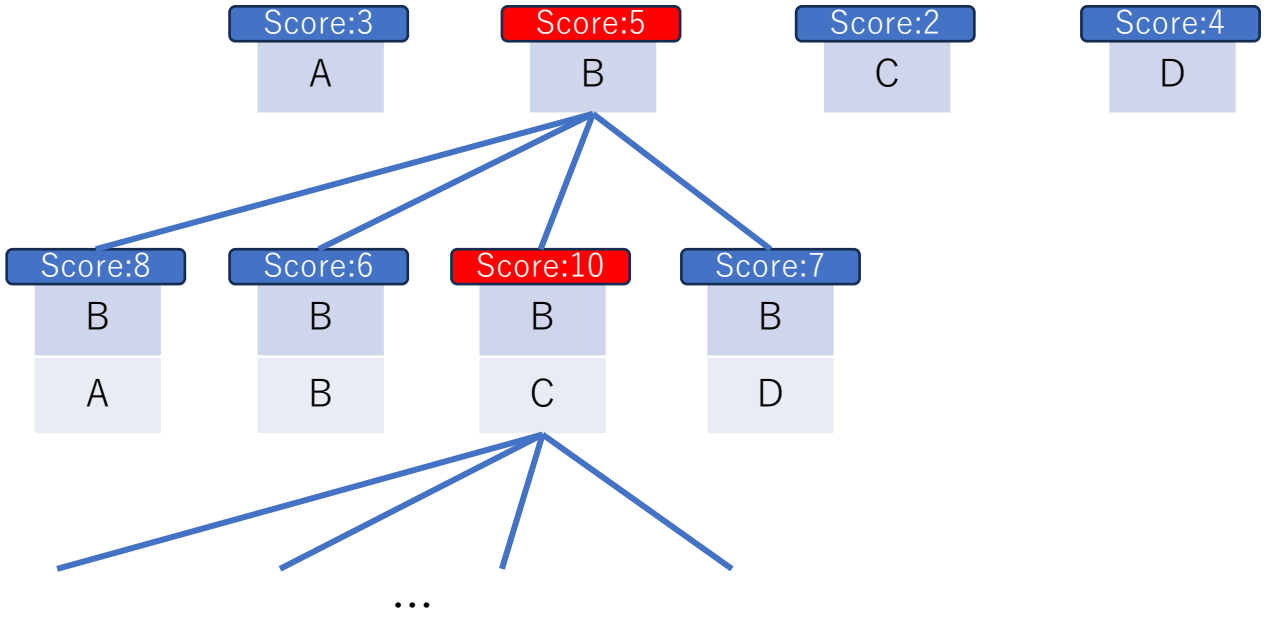
構築型
（貪欲、ビームサーチ）



解を評価しながら徐々に構築

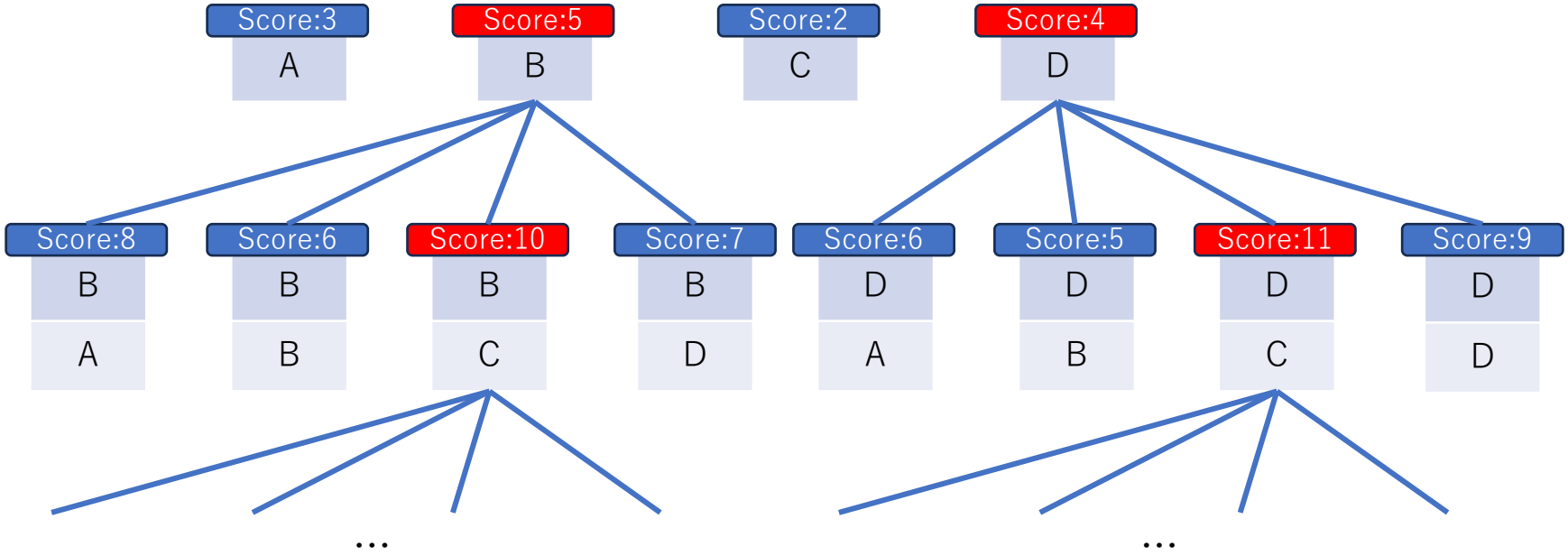
貪欲法

解を段階的に構築し、各段階で最も評価が高い1個を採用する



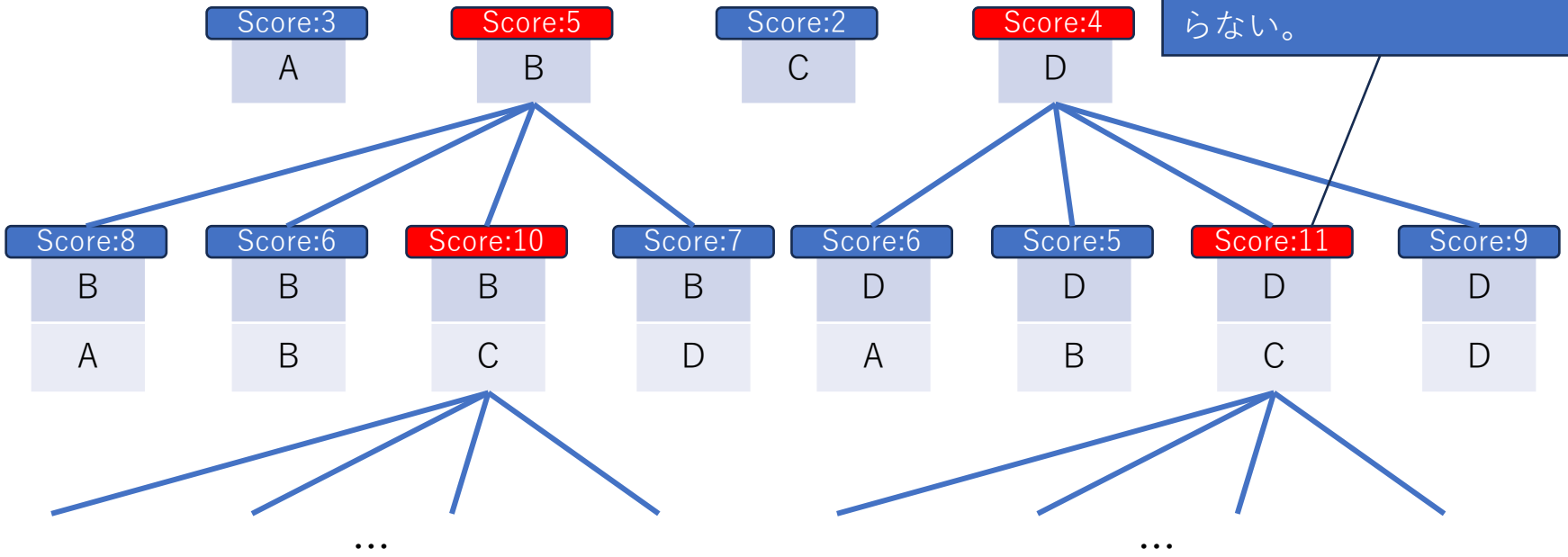
ビームサーチ 1/2

解を段階的に構築し、各段階で評価が上位W個のものを残す
※このWをビーム幅と呼びます



ビームサーチ 2/2

解を段階的に構築し、各段階で評価が上位W個のものを残す



1手目ではスコア2位だったDに続く解が、2手目ではスコア1位になった。
これは貪欲法では見つからない。

演習1

手を動かし、ビームサーチを使う

演習1: サンプルコードを完成させよう！ 1/12

必要なものを準備しよう

- 公式ツール
  https://img.atcoder.jp/intro-heuristics/tools.zip
- 公式ツール(exe版。WindowsかつWSLを使わない方のみ)
  https://img.atcoder.jp/intro-heuristics/tools_windows.zip
- サンプルコード
  https://img.atcoder.jp/ahf2-a7k3m9q2/c2b06a40b284660fb40cfac389d1a0e1.zip
- cargo(exe版を使用しない方)
  https://www.rust-lang.org/ja

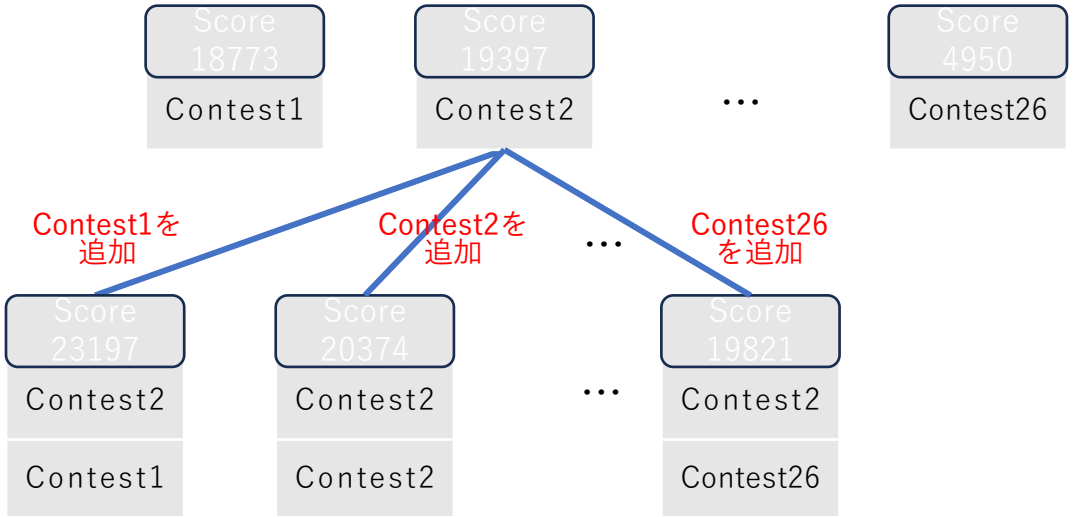
演習1: サンプルコードを完成させよう！ 2/12

以下の機能を持つクラスがあれば、貪欲法とビームサーチを簡単に使い分けできる

関数名	説明	AtCoder Contest Schedulingでの例
legalActions	現在の部分解を更新するための操作の候補一覧を取得する	「今見ている日付にコンテストを1種類埋める」を1操作として、26種類のコンテストを全て取得する
advance(action)	指定した操作(action)で解と評価値を更新する	今見ている日付に指定したコンテストを埋め、日付、満足度を更新する
isDone	解が完成したかどうかを判定する	365日目までコンテストを埋めたかを判定する

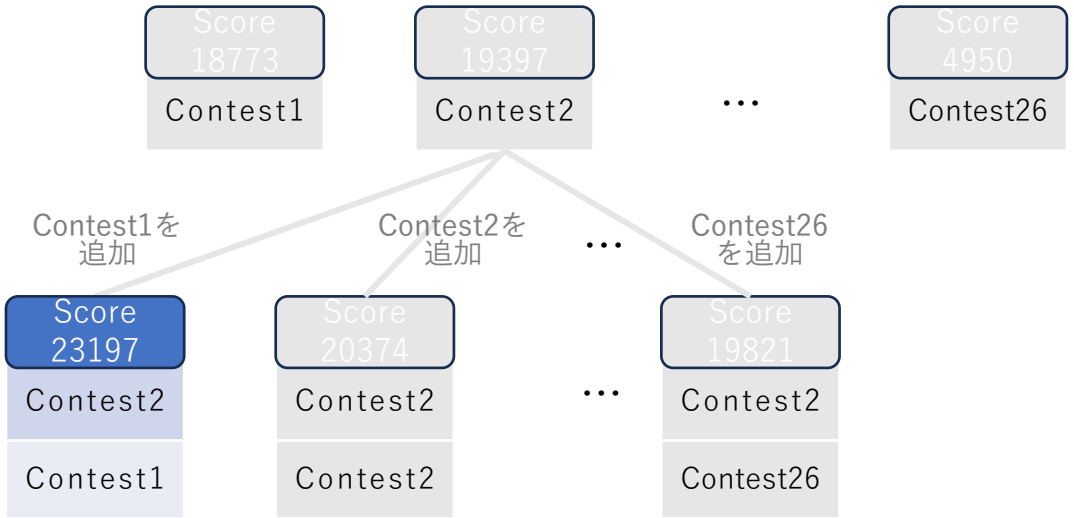
演習1: サンプルコードを完成させよう！ 3/12

legalActions: 現在の部分解を更新するための操作の候補一覧を取得する
※今回の実装では0-indexedで0,1...,25を返すことを想定



演習1: サンプルコードを完成させよう！ 4/12

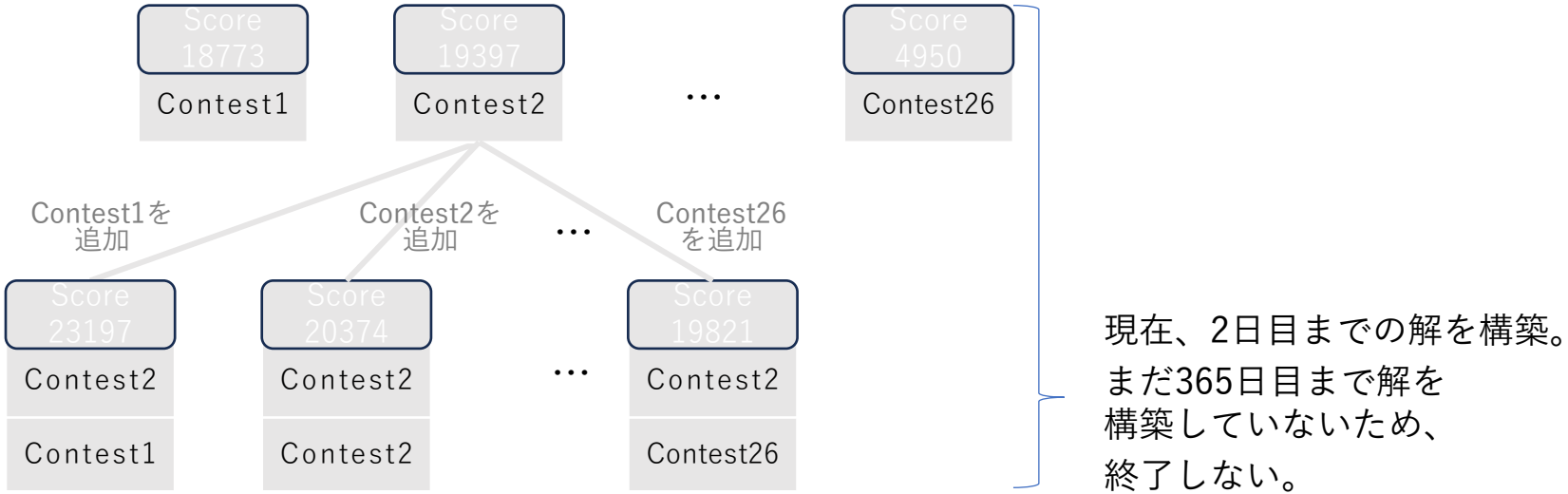
advance(action): 指定した操作(action)で解と評価値を更新する



2日目にContest1を開催したことによる満足度増加と、
開催されなかったコンテストの満足度減少、
日付を更新

演習1: サンプルコードを完成させよう！ 5/12

isDone:解が完成したかどうかを判定する



現在、2日目までの解を構築。
まだ365日目まで解を
構築していないため、
終了しない。

演習1: サンプルコードを完成させよう！ 6/12

サンプルコードをサポートページからダウンロードし、
コードが完成したら実行してみよう。
https://img.atcoder.jp/ahf2-a7k3m9q2/c2b06a40b284660fb40cfac389d1a0e1.zip

C++

```
# コンパイル

g++ -std=c++20 -O2 samples/cpp/sample_greedy.cpp -o a.out

# 実行

mkdir -p tools/out
./a.out tools/in/0000.txt tools/out/greedy.txt

# 評価

cd tools

cargo run -r --bin vis ./in/0000.txt ./out/greedy.txt
```

Python

```
# 実行

mkdir -p tools/out
python3 samples/python/sample_greedy.py <
tools/in/0000.txt > tools/out/greedy.txt

# 評価

cd tools

cargo run -r --bin vis ./in/0000.txt ./out/greedy.txt
```

演習1: サンプルコードを完成させよう！ 7/12

Windowsで非WSL環境の場合
https://img.atcoder.jp/intro-heuristics/tools_windows.zip
を解凍してtools_x86_64-pc-windows-gnuをサンプルコードのルートにコピー

C++

```
# コンパイル

自身の環境に応じてa.exeをつくる

# 実行
a.exe < "tools_x86_64-pc-windows-gnu¥in¥0000.txt" > "tools_x86_64-pc-windows-gnu¥out¥out0000.txt"

# 評価

"tools_x86_64-pc-windows-gnu¥vis.exe" "tools_x86_64-pc-windows-gnu¥in¥0000.txt" "tools_x86_64-pc-windows-gnu¥out¥out0000.txt"
```

Python

```
# 実行

python "samples¥python¥sample_beam.py" < "tools_x86_64-pc-windows-gnu¥in¥0000.txt" > "tools_x86_64-pc-windows-gnu¥out¥out0000.txt"

# 評価

"tools_x86_64-pc-windows-gnu¥vis.exe" "tools_x86_64-pc-windows-gnu¥in¥0000.txt" "tools_x86_64-pc-windows-gnu¥out¥out0000.txt"
```

演習1: サンプルコードを完成させよう！ 8/12

うまく実装できなければ、標準エラー出力を有効利用しよう
テスターへの出力を汚さずにデバッグができる

C++

```
int a=123;

cerr << a << endl;
```

Python

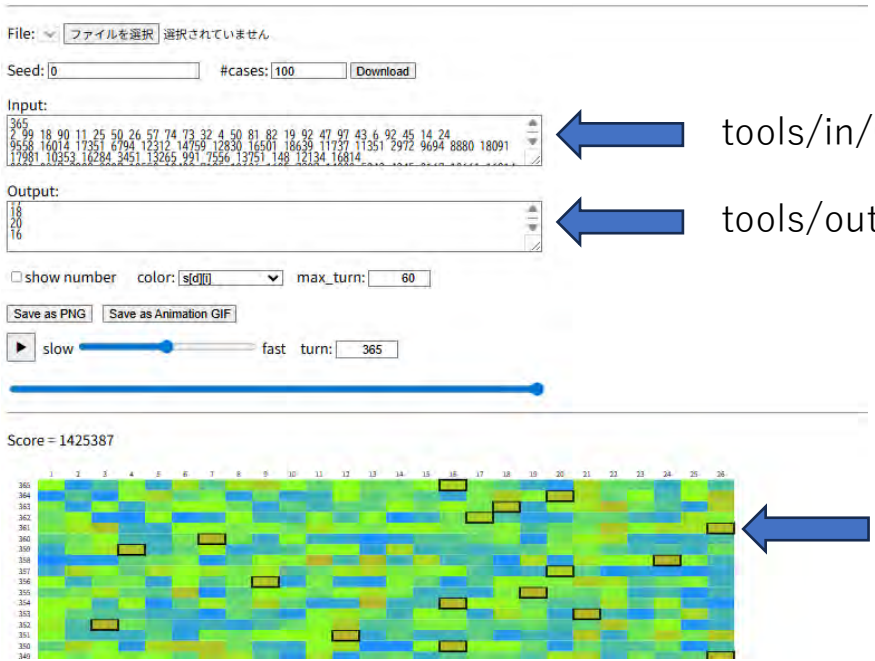
```
import sys

a=123

print(a,file=sys.stderr)
```

結果

```
123
```

演習1: サンプルコードを完成させよう！ 9/12

結果をビジュアライズしてみよう
リンク： https://img.atcoder.jp/intro-heuristics/vis.html?lang=ja



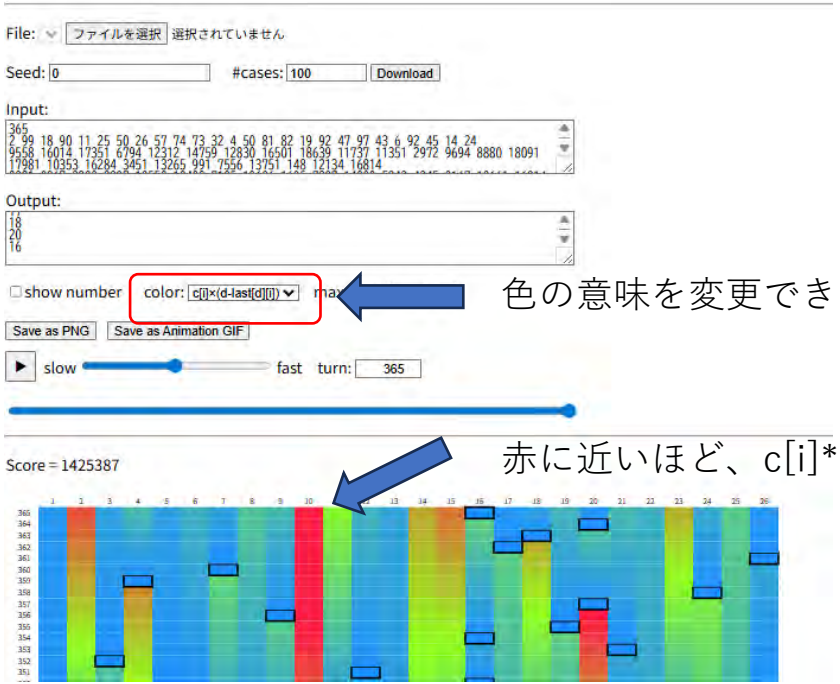
tools/in/0000.txt　と同様の文字列がすでに埋まってる

tools/out/greedy.txt の中身を張り付ける

ある日の選択したコンテストに枠がつく。
色が黄色寄りなほどs[d][i]が高い
(満足度が上がりやすい)

演習1: サンプルコードを完成させよう！ 10/12

結果をビジュアライズしてみよう
リンク： https://img.atcoder.jp/intro-heuristics/vis.html?lang=ja

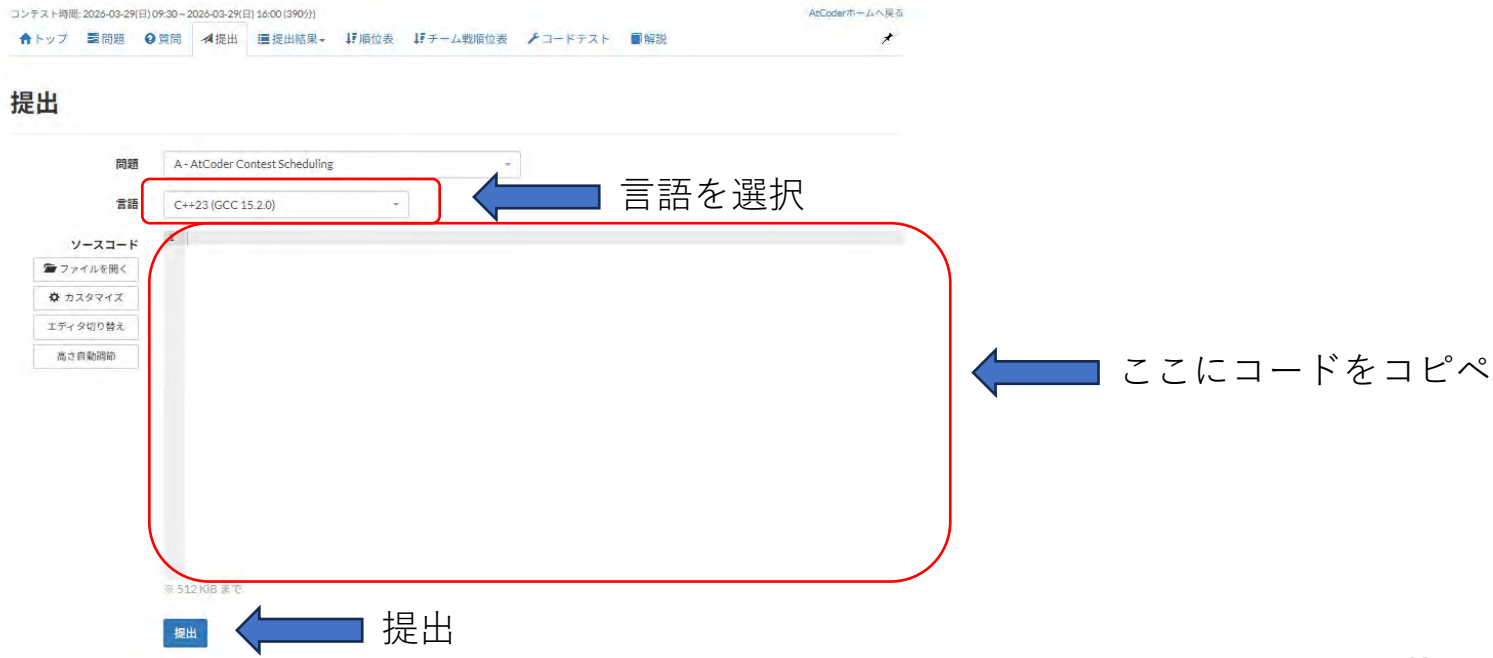


色の意味を変更できる。c[i]*(d-last[d][i])にもできる。

赤に近いほど、c[i]*(d-last[d][i])が高い。（満足度が減りやすい）

演習1: サンプルコードを完成させよう！ 11/12

コードを提出してみよう
リンク： https://atcoder.jp/contests/ahf2-a7k3m9q2/submit



演習1: サンプルコードを完成させよう！ 12/12

貪欲法で実装したlegalActions,isDone,advanceをsample_beam.cpp or pyにそのままコピーして、ビームサーチも同じように実行や提出をしよう

C++

```
# コンパイル

g++ -std=c++20 -O2 samples/cpp/sample_beam.cpp -o a.out

# 実行

./a.out tools/in/0000.txt tools/out/beam.txt

# 評価

cd tools

cargo run --bin vis tools/in/0000.txt tools/out/beam.txt
```

Python

```
# 実行



# 評価

cd tools

cargo run --bin vis tools/in/0000.txt tools/out/beam.txt
```

休憩

演習2

工夫し、ビームサーチを改善する

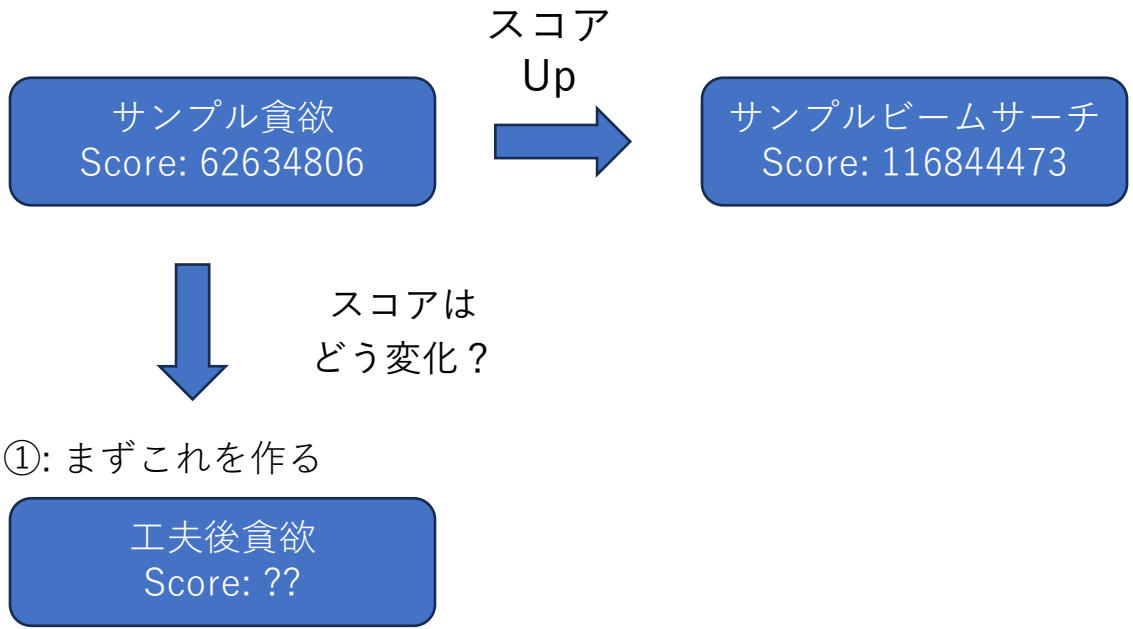
午後の部

午前中にどうしても自力でサンプルコードを完成できなかった方は、
以下のコードの中身をサンプルコードに移植しましょう。

```
.
├── answers
│   ├── cpp
│   │   ├── 01_greedy.cpp
│   │   └── 02_beam.cpp
│   └── python
│       ├── 01_greedy.py
│       └── 02_beam.py
```

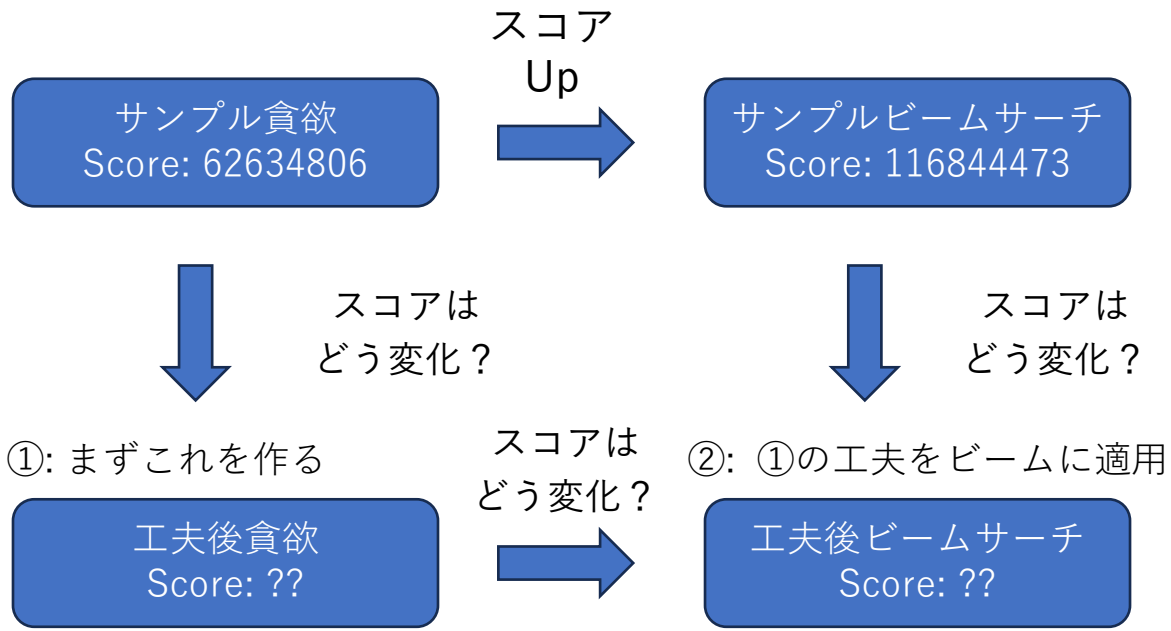
演習2: サンプルコードを改善しよう！ 1/12

サンプルコードに工夫を加え、スコアにどんな変化があるか、
提出して確認しましょう



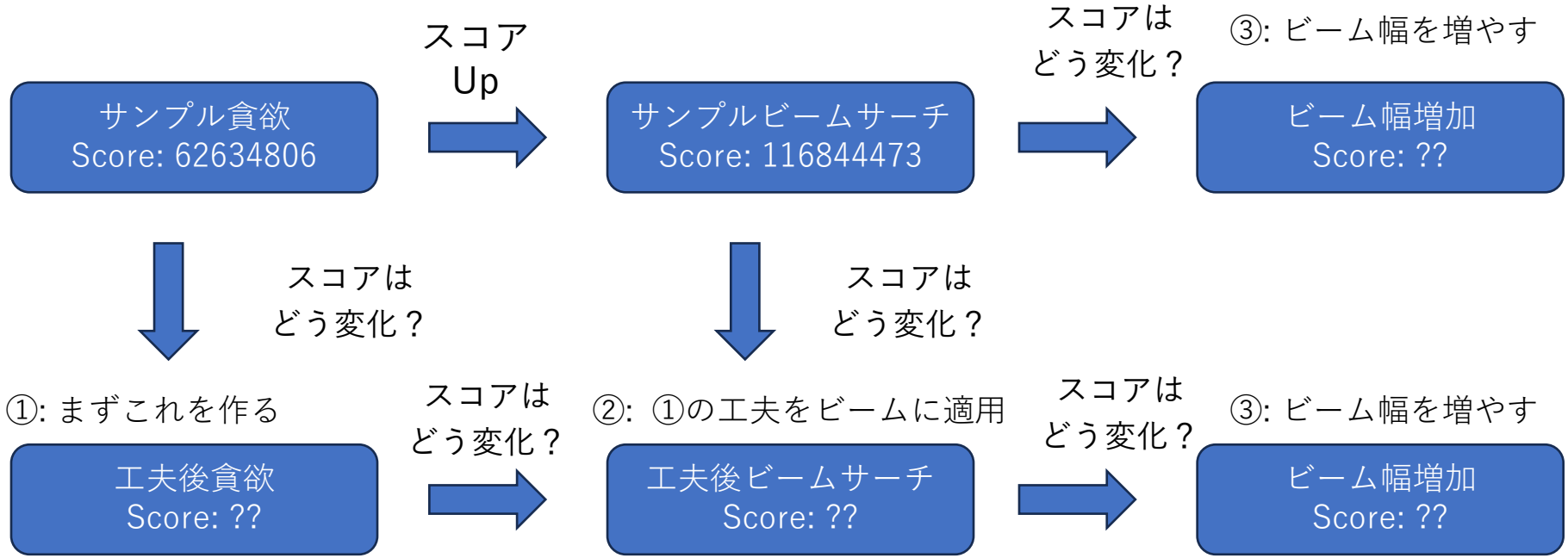
演習2: サンプルコードを改善しよう！ 2/12

サンプルコードに工夫を加え、スコアにどんな変化があるか、
提出して確認しましょう



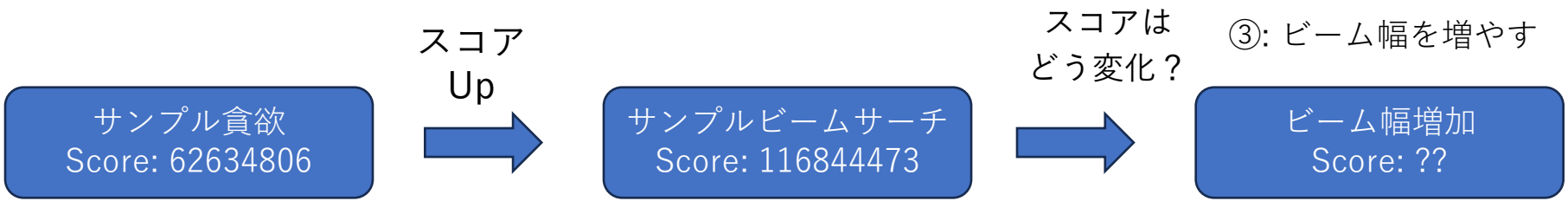
演習2: サンプルコードを改善しよう！ 3/12

サンプルコードに工夫を加え、スコアにどんな変化があるか、
提出して確認しましょう



演習2: サンプルコードを改善しよう！ 4/12

サンプルコードに工夫を加え、スコアにどんな変化があるか、
提出して確認しましょう



工夫方法がわからない人はこれだけでもやってみよう！

演習2: サンプルコードを改善しよう！ 5/12

では実装してみましょう
時間経過後にヒントを出します

演習2: サンプルコードを改善しよう！ 6/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

演習2: サンプルコードを改善しよう！ 7/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

追加ヒント: ・ビジュアライザでc[i]*(d-last[d][i])が赤い理由を考えよう
・高速化は考えず単純なループだけで実装する
・1つのコンテストで手計算し、
手計算と結果が合うか確認する
・将来は全てのコンテストを未開催としても評価できる
・とはいえ365日も未開催を続けることはないはず
・ということは程よく将来の評価をするといいかも

演習2: サンプルコードを改善しよう！ 8/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

ヒント2: legalActionsを修正し、評価対象を減らしてみる。
本当に毎回26種類のコンテストを試す必要ある？
その日と相性が悪かったり、開催直後だったりしない？

演習2: サンプルコードを改善しよう！ 9/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

ヒント2: legalActionsを修正し、評価対象を減らしてみる。
本当に毎回26種類のコンテストを試す必要ある？
その日と相性が悪かったり、開催直後だったりしない？

追加ヒント: ・ビジュアライザでs[d][i]を見てみる
・ビーム幅も増やせるかも

演習2: サンプルコードを改善しよう！ 10/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

ヒント2: legalActionsを修正し、評価対象を減らしてみる。
本当に毎回26種類のコンテストを試す必要ある？
その日と相性が悪かったり、開催直後だったりしない？

ヒント3: ヒント1の実行結果で違和感はないか？
ヒント1を発展できないか？

演習2: サンプルコードを改善しよう！ 11/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

ヒント2: legalActionsを修正し、評価対象を減らしてみる。
本当に毎回26種類のコンテストを試す必要ある？
その日と相性が悪かったり、開催直後だったりしない？

ヒント3: ヒント1の実行結果で違和感はないか？
ヒント1を発展できないか？

追加ヒント: 将来の満足度を計算するとき、
将来まで一切コンテストが
開催されていないとするのは不正確では？

演習2: サンプルコードを改善しよう！ 12/12

ヒント1: advanceを修正し、評価方法を変えてみる。
解の構築はd日目までしているが、本来は365日時点のスコアを最大化したい。
d日目時点の評価は365日目時点にとってうれしいか？

ヒント2: legalActionsを修正し、評価対象を減らしてみる。
本当に毎回26種類のコンテストを試す必要ある？
その日と相性が悪かったり、開催直後だったりしない？

ヒント3: ヒント1の実行結果で違和感はないか？
ヒント1を発展できないか？

ヒント4: 上記のヒントや自身で考えた工夫を複数組み合わせてみる

演習2: サンプルコードの改善例

後ほど公開

活用

ビームサーチを

今後活用するための考え方を説明

本当にビームサーチを使うか

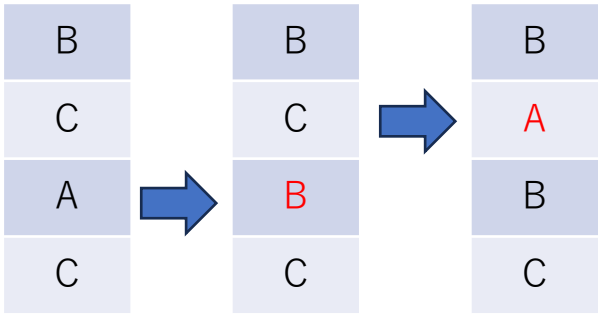
コンテスト1位もwriterも焼きなましを使用
AtCoder Contest Schedulingにおいて、ビームサーチは最適とは言い難い

手法	ユーザ	得点
焼きなまし	thunder	128176784
焼きなまし	wata(writer)	127939147
ビームサーチ	thunder	126072295
焼きなまし	shindannin（コンテスト1位）	125991044

局所探索型解法の特徴

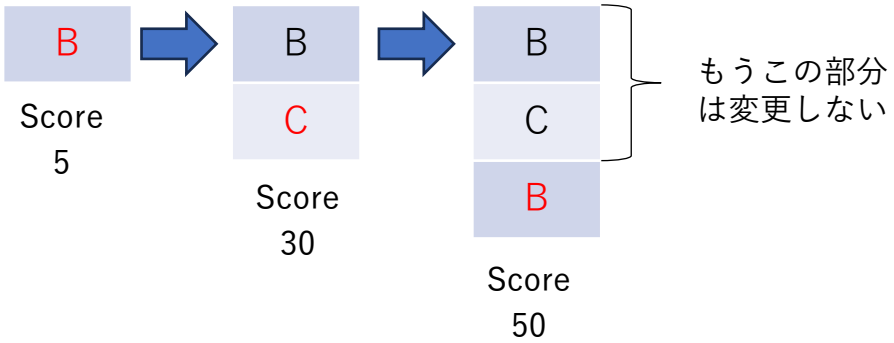
局所探索型解法の方が、
解の好きな部分を好きなタイミングで更新でき、探索の自由度が高い

局所探索型



いつでもどこでも変更可

構築型



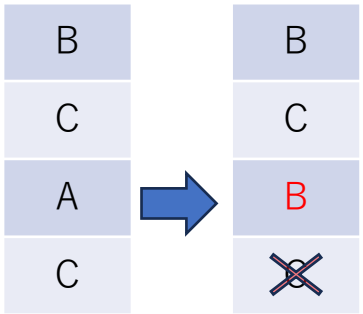
一度決定した部分を再構築しづらい

構築型解法の特徴

解の一部を変更すると、
解が成り立たなくなる or 他の部分に大きな影響を与える問題では有効

制約例
B,C,Aの次にCは許可されるが、
B,C,Bの次にCは許されない制約

局所探索型

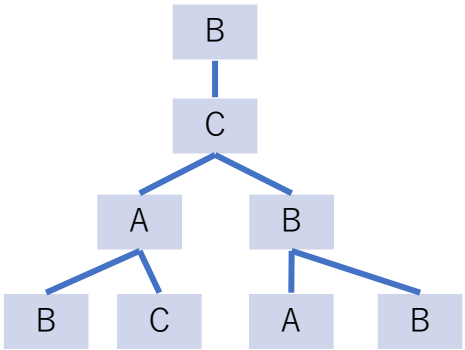


Score
50

WA

制約を満たす更新が難しい

構築型



制約を満たし、意味のある解だけを探索できる

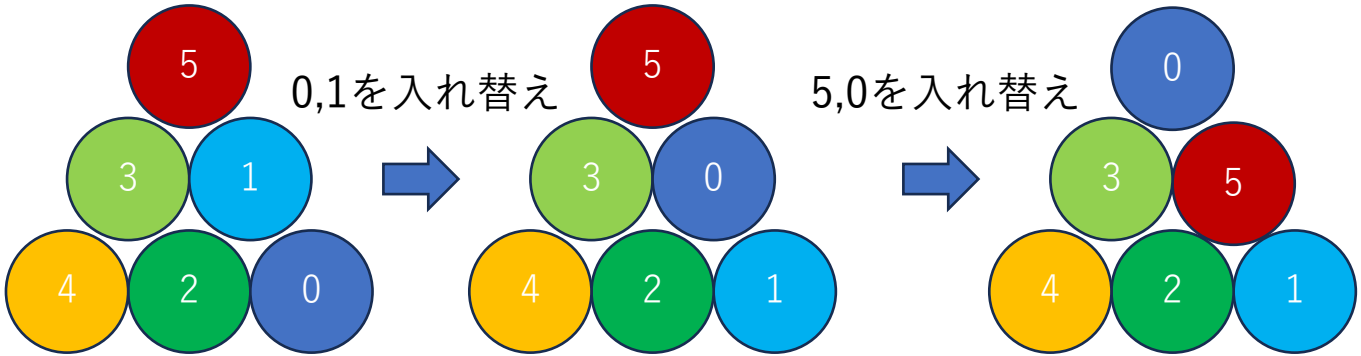
AtCoder Contest Schedulingでの有効性

局所探索型解法で困る要素が少なく、構築型解法の利点が活きない
→局所探索型解法が有利

解の変更に制約がない

構築型解法が活きる例

AHC021: Pyramid Sorting
隣合うボールを入れ替える操作を繰り返し、ボールの数字を上から順に並べる問題



事前に0,1を入れ替えていたから
5,0の入れ替えができた。
このような問題では、
構築型解法で段階的に解を作りたい。