11/31(土)

初期考察

- σはW,Hによらず一定なので、長いほど得.
  - 普通の正規分布の和なら $\sqrt{n\sigma^2}$ になるため.
- すると一列に並べたい気持ちになる.
  - 縦と横で2つ計測できる.
- 「変数集合を選ぶとその和が誤差付きで得られます.
  各変数の値を推定してください」という問題になる.
  - これはほぼAHC003
- 長方形詰め込みはがんばる.
  - 詰め込みは正規分布のような良い性質がなく離散的.
  - 正規分布から長方形を多数サンプリングして
    モンテカルロ的に詰め込み結果の期待値求めてもいいかも.

AHC003 + TOYOTA HC 2023 SPRING

12/1(日)

- 正規分布モデル.
  - 全ての w_i, h_i をまとめて X とおく.
  - X の事前分布.(正規分布で大胆近似).
    - 期待値 $x_0 = [65000, 65000, \dots]^T$
    - 分散共分散 $\Sigma_0 = \mathrm{diag}(21280^2, \dots)$
      - ↳ パラメータはモンテカルロで求めた.

- 1変数モデルのベイズ推定.
  - 計測誤差の分散 σ^2 は既知である.
  - データとして $\tilde{y}$ が得られたとき,事後分布は.
    - 平均: $\dfrac{\Sigma_0 \tilde{y} + \sigma^2 x_0}{\Sigma_0 + \sigma^2}$
    - 分散: $\dfrac{\Sigma_0 \sigma^2}{\Sigma_0 + \sigma^2}$

・多変数モデルのベイズ推定.

- クエリごとに長さ $y_t \sim N(h_t, \sigma^2)$ を得る.

  ここで、使った辺を1, 使わなかった辺を0とするベクトル c_t を用いて

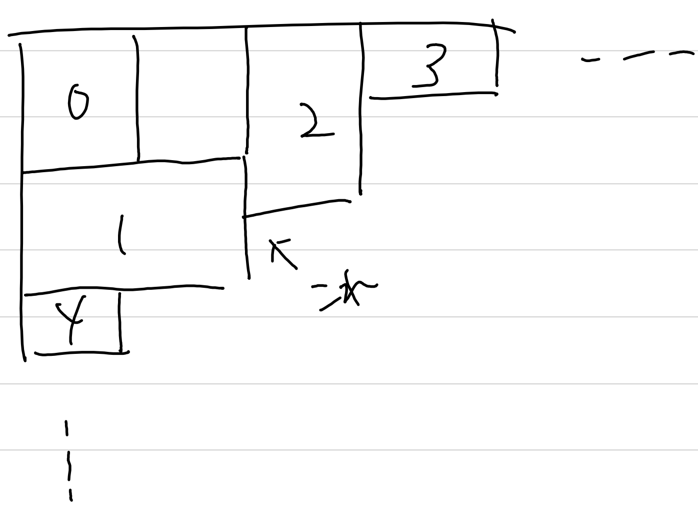
  $h_t = c_t^T X$

- このときの事後分布 $p(X=x \mid y_1, y_2, \cdots y_t) = N(x_t, \Sigma_t)$ は
  - $x_t = x_{t-1} + (S_{xy})_t (S_{yy})_t^{-1} (y_t - c_t^T x_{t-1})$
  - $\Sigma_t = \Sigma_{t-1} - (S_{xy})_t (S_{yy})_t^{-1} (S_{yx})_t$

- ここで、
  - $(S_{yx})_t = c_t^T \Sigma_{t-1}$ (横ベクトル)
  - $(S_{xy})_t = (S_{yx})_t^T$ (縦ベクトル)
  - $(S_{yy})_t = c_t^T \Sigma_{t-1} c_t + \sigma^2$ (スカラー)

・縦横の並べ方.

- とりあえずランダムで良い
- 直接接触しないケースに注意.



- パッキング
  - とりあえずビムサ
  - 使う長方形は多変量正規分布からランダムサンプリング
    - 複数の平行世界を同時に実行
    - 8個, 16個でSIMDをがんばる？単純な処理なら可
  - 遷移は全部試す
    - bottom-leftでないものはやらない
    - 他と被る遷移はやらない
      - 並行世界があるとムズい
      - 壁際を除くくらいか
  - 計算量
    - 愚直だと $O(N^2)$、スライド最小値なら $O(N \log N)$ か
      ソートがボトルネック
    - ビーム幅W, 並行世界K
    - 幅W × 世界K × ターンN × 各遷移 N^2
      → $O(WKN^3)$ まあまあ重い
  - 一旦並行世界なしでやる
  - スコア
    - とりあえずW+Hにする？

- パッキング
  - 左方向に入れたとき、上側がピッタリくっついていないなら
    ダメそうと判断して遷移しない
    - 長方形たちとx座標が合っているかで判断できる
  - パッキング中は計測データを捨てる
    - もったいないのでなんとかしたい
  - 一旦実装
    - ビーム幅はNにかなり左右される
    - 47.86Gで50位
  - 思ったより大きかったブロックに引っかかってうまくパッキング
    できていない
    - 近すぎたらペナルティとしてバッファを設ける
    - 49.01Gで8位
      - やっぱりロバスト性は大事
      - 1位はchokudaiさんで49.75G
  - ビーム幅を増やしてもそこまで改善しなさそう
    - せいぜい1%くらい
  - 推定精度が上がるとスコアに寄与しそう
    - クエリ数2倍で+1.3%、5倍で+2%
    - 推定精度を上げるか、結果を有効活用するか

計測時の
組合せをいい感じに?

- 明日やること
  - 並行世界でSIMD
    - 座標に32bitもいらないので、丸めて16bitにしたい

12/2(月)

今日は実装Day

- 並行世界の実装
  - N個の長方形のセットを16セット作り、
    それぞれを1世界として16世界を同時に動かすビームサ
  - まずはSIMDなしで実装
    - SIMD処理しやすい形にするため
      データの持ち方を変える必要がある
    - 配列の次元順が非直観的でバグらせる…
    - 隣接長方形とのバッファは
      並行世界シミュレーションで代用できるので削除する
    - 実装、悪化はしていないが当然重い
  - SIMD化
    - AVX2をゴリゴリ使う
      - AVX512はnightly Rustでないと使えない(悲しい…)
      - まあ2倍速にはならないはず
    - ChatGPTが優秀
    - 座標は$\frac{1}{2^6}$倍して16bit整数で扱う
    - バグらせず一発で動いてビビる
    - スコアは全並行世界の高さ×幅の和
    - やや改善して、スコア+0.3%くらい

- クエリは何度も発行できるので、
  期待値を上げるよりは上振れ狙いの方が良い。
- スコアを全世界の和ではなく
  上位8世界の和にした。
- これは当たって、さらにスコアが +0.5%
- 49,346Gで3位。

- 実験
  - ビーム幅10倍 → スコア +0.1%
  - クエリ数2倍 → スコア +0.8%
  - クエリ数5倍 → スコア +1.1%

  ⇓

  辺の長さの推定精度を上げることが重要。

- やりたいこと
  - 推定分散を最小化する箱の置き方。
  - 長さ調査パートだけでなく、箱詰め解答パートでも
    クエリの結果を有効活用する。
  - ビームサでアスペクト比を考慮していない問題の修正

12/3 (火)

・情報量

- eivourさんのAHC003記事を参考に
  情報量の大きい箱の並べ方を考える.
- 分散共分散行列Σの固有値(の平方根)は行列のスケーリングと
  捉えることができるので これが減ると嬉しい.
- 行列の対角成分の和をトレースという.
  tr(A)はAの固有値の和と等しい.
- よってtr(Σ)が小さくなると
  多変量正規分布の裾野が小さくなって嬉しい. (たぶん)
- 分散共分散行列の更新式は
  $$\Sigma_t = \Sigma_{t-1} - (S_{xy})_t (S_{yy})_t^{-1} (S_{yx})_t$$
- $tr(\Sigma_t) = tr(\Sigma_{t-1}) - tr((S_{xy})_t (S_{yy})_t^{-1} (S_{yx})_t)$ なので.
  $tr((S_{xy})_t (S_{yy})_t^{-1} (S_{yx}))$ を大きくしたい.
- $(S_{xy})_t = \Sigma_{t-1} C_t$, $(S_{yx})_t = (S_{xy})_t^T$,
  $(S_{yy})_t = C_t^T \Sigma C_t + \sigma^2$ より.
  $$tr((S_{xy})_t (S_{yy})_t^{-1} (S_{yx})_t) = tr\left(\frac{\Sigma_{t-1} C_t C_t^T \Sigma_{t-1}}{C_t^T \Sigma_{t-1} C_t + \sigma^2}\right)$$
  $$= \frac{\|\Sigma_{t-1} C_t\|^2}{C_t^T \Sigma_{t-1} C_t + \sigma^2}$$
- 確かに同じ式になった.

・差分計算.

・$C_{\pi,i}^{T}$ が ΔC だけ変化したときのトレースの変化量を考える.

・(分母の変化量) $= \begin{pmatrix} 0 & 0 & \cdots & \Delta C & \cdots & 0 \end{pmatrix} \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1N} \\ \vdots & \ddots & \\ \sigma_{N1} & & \sigma_{NN} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ \Delta C_i \\ \vdots \\ 0 \end{pmatrix}$

$$= \begin{pmatrix} 0 & 0 & \cdots & \Delta C & \cdots & 0 \end{pmatrix} \begin{pmatrix} \sigma_{1i} \\ \vdots \\ \sigma_{Ni} \end{pmatrix} \Delta C_i$$

$$= \sigma_{ii} (\Delta C_i)^2$$

・(分子の変化量) $= \left\| \begin{pmatrix} \sigma_{1i} \\ \vdots \\ \sigma_{Ni} \end{pmatrix} \Delta C_i \right\|^2$

$$= (\Delta C_i)^2 \sum_{j=1}^{N} \sigma_{ij}^2$$

・よって, $\mathrm{tr}(\cdots) = \dfrac{(\Delta C_i)^2 \sum_{j=1}^{N} \sigma_{ij}^2}{\sigma_{ii} (\Delta C_i)^2 + \sigma^2}$

・1変数で $\Delta C_i = 1$ のときを考えると, $\sigma_{11} = \sigma_\pi^2$ に注意して,

$$\frac{\sigma_\pi^4}{\sigma_\pi^2 + \sigma^2} = \frac{\sigma_\pi^2 (\sigma_\pi^2 + \sigma^2) - \sigma_\pi^2 \sigma^2}{\sigma_\pi^2 + \sigma^2}$$

$$= \sigma_\pi^2 - \frac{\sigma_\pi^2 \sigma^2}{\sigma_\pi^2 + \sigma^2} \quad \rightarrow \text{正しそう.}$$

・$\sum_{j=1}^{N} \sigma_{ij}^2$ を前計算しておけば, 差分計算は $O(1)$ で可能.

$$(c_i+\Delta c)^2 - c_i^2 = 2c_i\Delta c + \Delta c_i^2$$

・差分計算

- いろいろ嘘だったのでやり直し.
- (分母) $= (c_1 \cdots c_i+\Delta c \cdots c_N)\, \Sigma \begin{pmatrix} c_1 \\ \vdots \\ c_i+\Delta c \\ \vdots \\ c_N \end{pmatrix}$
- (分母の変化量) $= 2\Delta c \sum_{j=1}^{N} \sigma_{ij} c_j + \sigma_{ii} \Delta c_i^2$
- (分子) $= \left\| \Sigma \begin{pmatrix} c_1 \\ \vdots \\ c_i+\Delta c \\ \vdots \\ c_N \end{pmatrix} \right\|^2$

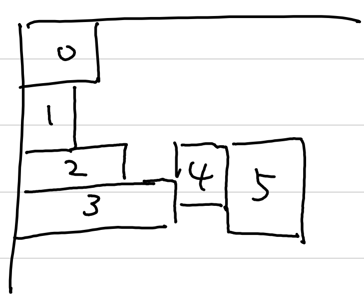
$$= \left\| \begin{pmatrix} \sigma_{11} c_1 + \cdots + \sigma_{1i}(c_i+\Delta c) + \cdots \sigma_{NN} c_N \\ \vdots \\ \vdots \end{pmatrix} \right\|^2$$

$$= \left\| \begin{pmatrix} (\sigma_{11} c_1 + \cdots + \sigma_{1i} c_i + \cdots + \sigma_{NN} c_N) + \sigma_{1i}\Delta c \\ \vdots \\ \vdots \end{pmatrix} \right\|^2$$

- (分子の変化量) $= 2\Delta c \sum_{j=1}^{N} \left(\sigma_{ji} \underbrace{\sum_{k=1}^{N} \sigma_{jk} c_k}_{\Sigma c} \right) + \Delta c^2 \sum_{j=1}^{N} \sigma_{ji}^2$

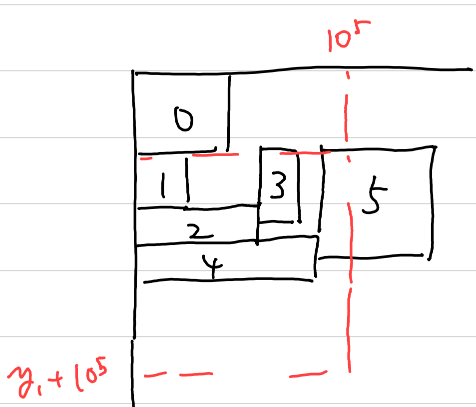
- これは $\sum_{k=1}^{N} \sigma_{jk} c_k$ を各jに対して保持しておけば
  O(N)で計算可能.
  - 更新時は $S_j' = S_j + \sigma_{ji}\Delta c$ とすればよい.

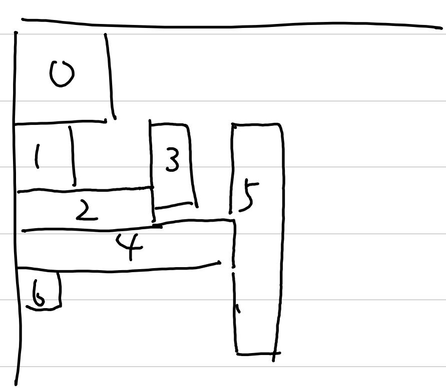
- 解の持ち方
  - 1次元なら簡単だが、2次元だとややこしい。
  - 各箱はVertical, Horizontal, Noneの3状態を持つ。
  - 箱0はVertical固定。
  - ある1つの箱がpivotとなる。
    - 最初のHorizontalまでのどれかがpivot.
    - Horizontalの箱はpivot−1を目指して置く。



pivot=2の例.

  - rotateするかどうかのboolも持つ。
  - 引っかかるかどうか判定。





  - 10^5を超えれば大丈夫。
  - ~~Verticalの場合~~。　　Stackではうまくいかない。
    - ~~(x_0, y_1)をstackに入れていく。(x_0昇順)~~
    - ~~配置時にstackをtopから見ていき、初めに当たった($x_0<w$) y_1に配置~~。

・スコア.
  ・tr(縦) + tr(横) を2回観測したときの固有値減少量
    だが、真面目にやると計算量がヤバいので妥協.

・実装.
  ・つらい
  ・配置のシミュレーションで死ぬほどバグった.
  ・確かに分散は小さくなったが、スコアはほぼ不変. そんな….

- 領域を正方形に近付けたい。
  - 最初から幅を $\sqrt{\sum h_i w_i}$ だけ確保しておき、
    超えた場合は重めのペナルティをかける。
  - なぜかペナルティの大きい方に伸びていく。なぜ…。
  - ペナルティ値を元に戻したら普通に動いた。それで良いんかい。
  - 実装一瞬な割にそこそこ伸びた(+0.25%)。
    なんだか複雑な気分に。
  - 49.33Gで4位。

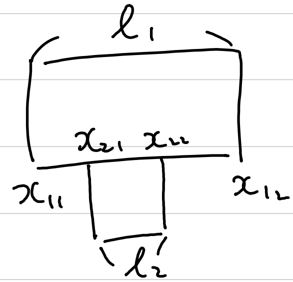
12/4(水)

- 推定精度実験
  - テストケースのσを変えて実験.
  - σを2/3にするとスコア+0.6%、1/10なら+1.2%
  - 昨日の結果から精度は影響が小さいと思っていたが
    そこそこ効果がありそう.

- 推定器の改良.
  - ビームサーチ中に捨てている情報を有効活用したい.
  - 今考えている方針は2つ.
    - 事前にたくさん生成しておき,対数尤度の高いものを採用…①
    - 焼きなましで対数尤度の高いものを求める. …②
  - 計算量.
    ① 生成数をK個として、クエリごとに $O(KN^2)$
    ② 焼きなましの遷移ごとに $O(TN^2)$
  - ②の方が良い結果になりそうだが,計算量が重い.
    まずは①からやる.
  - 初期生成は今まで通り多変量ガウス分布でやる.

- 実装
  - 頑張って実装した…が、誤差の範囲
    - 0.05%くらい良くなった…かも、くらい
    - うーん…
  - ちなみにこれ入れたらサーバ上で1ケースだけTLEした
    - $N \sim 100, T \sim 400$ だとかなり重いっぽい
    - その場合は多変量正規分布のベイズ推定に切り替える

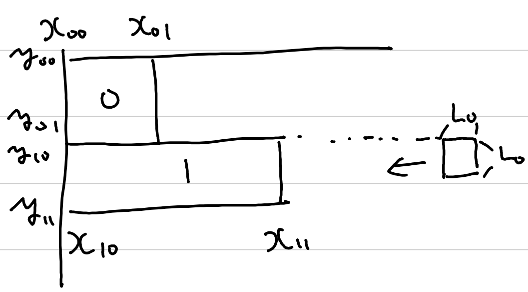
- 「ピッタリくっついてる」判定の改善
  - 「ちょっと引っかかってる」だけのやつは除きたい
  - $\min(x_{12}, x_{22}) - \max(x_{11}, x_{21}) \geq threshold$ で判定可
  - $threshold = \min(l_1, l_2) \div 4$ とした
    - 2のべき乗で割るのはビットシフトで表現できる
  - 0.05%くらい伸びた
    - 地道すぎる…



12/5(木)

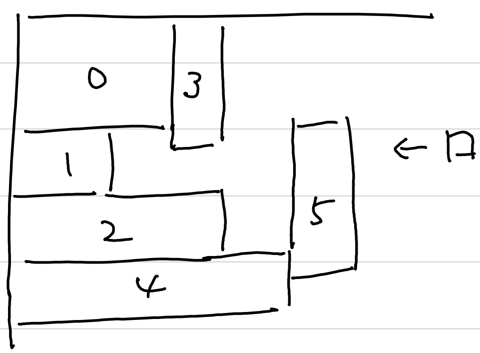
- 箱配置の計算量改善
  - MCTSなども試してみたいが、計算量が絶望的
  - 今の計算量
    - 配置する箱の数が $\Theta(N)$
    - 1回ごとの配置候補が $\Theta(N)$
    - 配置候補あたりの衝突判定が $\Theta(N)$
    - $\longrightarrow$ 全部で $\Theta(N^3)$ (やばい)
  - ほとんどの配置候補は無駄なはず

- 配置候補数の削減
  - 下みたいな配置となった場合、長方形0を試すのは無駄



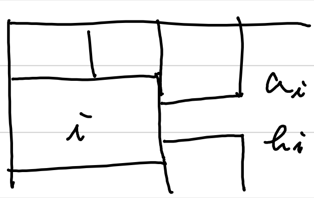
  - 無駄は言いすぎだが、良い手である見込みは薄い
    - 枝刈りすることで効率化できるはず
  - 判定方法
    - 最小の箱の大きさを $L_0 = 20000$ とする
    - $\max(y_{i1}, y_{j0}) < \min(y_{i1} + L_0, y_{j1})$ && $x_{i1} < x_{j1}$ なら無駄
  - 計算量
    - 有効な候補数は平均 $O(\sqrt{N})$ 程度になるはず
    - それぞれに対して衝突判定が $O(N)$ なので、
      全体で平均 $O(N^2\sqrt{N})$

- 衝突判定の計算量削減
  - 以下のようなケースでは箱1との衝突を気にしなくてよい.



※ baseとピッタリくっつける
という制約を課している為.
本来5はここに置かれない.

- 判定
  - 箱 i の露出長さが L_0 未満なら衝突判定不要.
  - 露出座標を a_i, b_i とする.



  - 箱が宙に浮くことがないことに注意すると,
    箱 j を昇順に見ていき,

$$a_i \leftarrow \max\left(a_i, \begin{cases} y_{j1} & \text{if } x_{i1} < x_{j1} \text{ and } y_{j0} < y_{i1} \\ 0 & \text{otherwise} \end{cases}\right).$$

$$b_i \leftarrow \min\left(b_i, \begin{cases} y_{j0} & \text{if } x_{i1} < x_{j1} \text{ and } y_{j1} > y_{i0} \\ \text{inf} & \text{otherwise} \end{cases}\right)$$

- 計算量.
  - これも平均 $O(\sqrt{N})$ になるはず.
  - 配置候補数が $O(\sqrt{N})$, 判定に $O(\sqrt{N})$ なので, 1遷移は平均 $O(N)$
  - 更新処理も適切に行えば全体 $O(N^2)$ になるはず? → ならない.

- ソートの高速化
  - スコア計算のために16世界のスコアをソートしている.
  - 並列数をPとすると. ソートは $\Theta(P\log_2 P)$
    - よって全体の計算量は平均 $\Theta(N^2+N\sqrt{H}P\log_2 P)$
  - ソートがボトルネックになる可能性があるため,
    これも高速化したい.
  - いろいろ調べると, バイトニックソートというアルゴリズムがあった.
    - 比較回数 $\Theta(P(\log_2 P)^2)$
    - ただし各比較は並列処理可能で,
      比較のステップ数は $\Theta((\log_2 P)^2)$ となる.
    - 16要素のソートなら10ステップで終わる(すごい)
  - また, P要素の水平加算は $O(\log_2 P)$ でできる.
  - よって全体の計算量は $\Theta(N^2+N\sqrt{H}(\log_2 P)^2)$

てか考えてたけで
デスノートに全てを破壊されてしまった…。

12/6 (金)

- お仕事お休みにしてAHC

- 配置判定数削減
  - 頑張って実装。
  - seed=1で10倍くらい速くなった(すごい)
  - 候補数は10個くらいに抑えられていそう。
  - 49.28Gで4位。

- 衝突判定削減
  - うーん微妙。
  - 意外と減らせないみたいなのでやらなくていいかも。

- バイトニックソート
  - 本当にソートされた。すげー!
  - アセンブリ見ると vpshufd とか vpblendd とか いっぱい出てきて壮観
  - 速くなったかは怪しい……。
  - でも楽しかったのでOK.

- 配置判定数削減.
  - もっと候補数削れそう.
  - 最初から幅を $\sqrt{\sum A_i}$ だけ確保しているので.
    これを超えてほしくない.

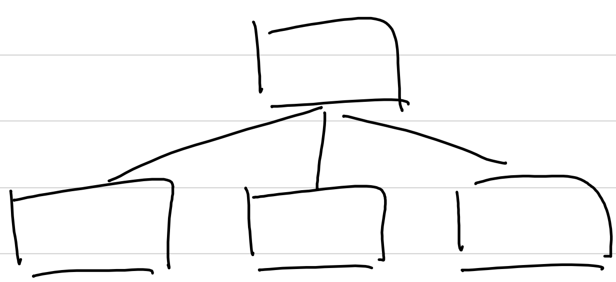
  - 超える場合はNGとする.
  - かなり速くなった.

- 高速化結果.
  - seed = 1 (N = 87) でビーム幅が10〜15倍になった.
  - スコアが0.4%くらい伸びた.
  - MCTS試さなきゃ….
    - 候補数がかなり絞られているのでやりやすいはず.
  - 焼きなましでサンプリングするやつもやりたい.

12/17(土)

- MCTS
  - ビームサーチの評価値を「その時点の最大W+H」とするのはあまり良くない。
    - プレイアウトなら評価しやすそう → MCTS
  - ノードの情報
    - 試行回数、累計評価値
    - 木上のビームサーチのSmallState
    - 木上のビームサーチのLargeState

    ビームサーチも木でやってるので情報としてはほぼ同じはず。
  - 候補の数
    - UCB1計算時に並列計算したい。
    - あまり無駄な候補を増やしたくない。
      - 最大8候補くらいにしたい。
      - sqrtもlnもargmaxもSIMDで表現できるはず。
- MCTSおさらい



選択
↓
評価
↓
展開
↓
更新(逆伝搬)

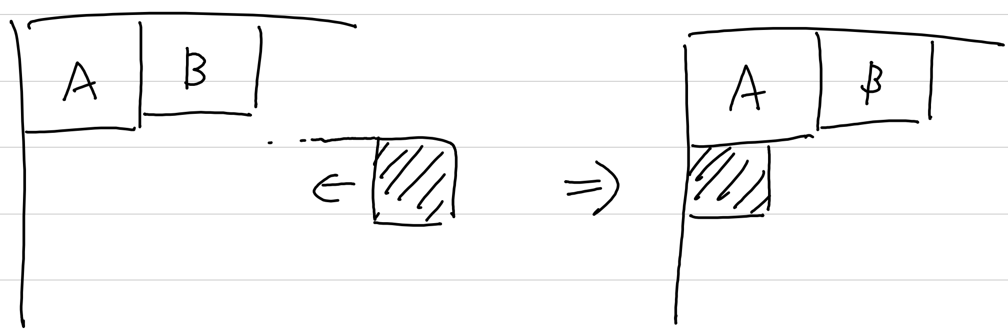
- MCTSつづき.
  - 候補手の絞り方.
    - 候補が8個あるので、回転ありなしで4個ずつにしたい.
    - その上でBottom-Left順に4つずつ取る.
  - 実装
    - バグらせながらなんとか.
    - +にすべきところを×にしていたりでひどい.
  - 性能
    - MCTS単体だと弱い.
    - 中盤までビームサ、終盤MCTSとやると改善した
      - 0.2%くらい改善.
- 投入方向.
  - 今まで右から、下から両方から箱を投入していたが、
    下からだけで充分そう.
  - 0.1%くらい改善
- MCTSプレイアウト.
  - Bottom-LeftよりTopが低くなることを優先した方が良い.
  - ここまで入れて49.40G、実質5位くらい.

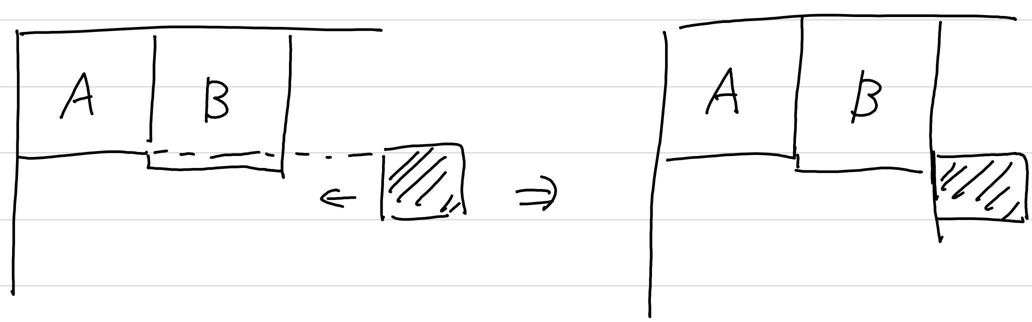
大きさの差が
微妙な場合に
横から入れるのが本質?

12/8(日)

- MCMC
  - まずはMCMCを実装していく.
  - 今までと同様にAVX/AVX2を使うことで
    8並列/16並列でMCMCが可能.
  - なんか上手く推定できないと思ってたら
    長方形の辺の長さの下限が2×10^4だと勘違いしていた….
  - そして最初に与えられる計測も無視していた. ひどい….
  - 修正したらちゃんと推定できるようになった.
    - 0.1%くらい改善.
    - 1イテレーションに30μsくらい.
  - 提出したがさほど変わらず.

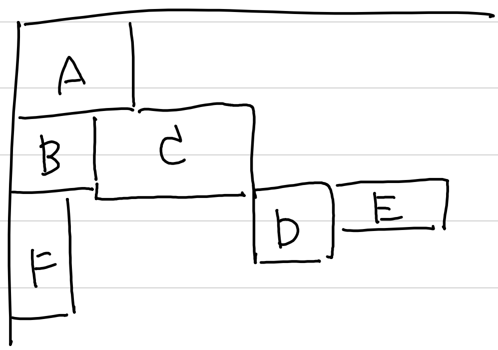
- 大小比較
  - うまく横から挿入することで大小比較が可能



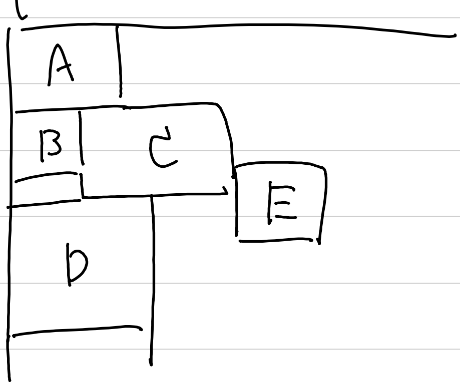


  - 引っかかるか引っかからないかでかなり大きな情報が取れる
    - もっと早く気付くべきだった

・大小比較方針
  ・大きさがほぼ同じやつを2つ並べる.
  ・そして小さい方に合わせて投入



  ・上の例ではBをbaseとして横からDを投入.
  ・複数個作ることもできる?
    ・色々大変そう.
    ・一旦T字形で考える
- 条件



  - A〜B間は縦or無のみ.
  - B〜C間は高さが十分小さくないとNG.
  ・B+Cの幅はDの幅より十分大きい.
  ・200000×200000まで衝突可能性あり.

- 選択方針
  - 2つの正規分布 $N(\mu_1, \sigma_1^2)$, $N(\mu_2, \sigma_2^2)$ $(\mu_1 \leq \mu_2)$
    について、大小関係が入れ変わる確率は、
    正規分布 $N(0, \sigma_1^2+\sigma_2^2)$ で $x \leq \mu_1 - \mu_2$ となる確率.

$$\Rightarrow \frac{1}{2}\left(1+\mathrm{erf}\frac{\mu_1-\mu_2}{\sqrt{2(\sigma_1^2+\sigma_2^2)}}\right)$$

- 実装
  - むずすぎ…
  - 実装後もずっとバグ取り.
  - バグは取れた気がするが.
    対数尤度がとんでもない値に…
    - $w_i < w_j$ を満たすか満たさないかで
      非連続的に対数尤度が変わるため
      一度サンプリングが迷子になると帰ってこられない?

・パラメータチューニング
・いつも通り以下の手順
① $x_i = (N_i, T_i, \sigma_i)$ をランダムサンプリング
② サンプルした点でoptunaを使い
最適パラメータベクトル y_i を得る.
③ ①,②を繰り返して X, Y を得て,
ガウス過程回帰により補間関数 $y = f(x)$ を得る.
・88コアインスタンスを借りて回して寝る.

12/9 (月)

- パラメータチューニング
  - 起きたらある程度結果が集まっていた.
  - ガウス過程回帰のコードを書く.

    ↓
  - いろいろやってたらチューニング結果ファイルを消してしまった (は?)
    - 終わった….
    - 88コア×2台に変更. 間に合ってくれ….

- 大小比較のやつ.
  - いろいろやったが上手くいかない.
  - 締め….

- パラメータ
  - どうにか間に合った.
  - 0.2%くらい改善. これで最後かな.

    → 49,193G, 暫定6位で終了.