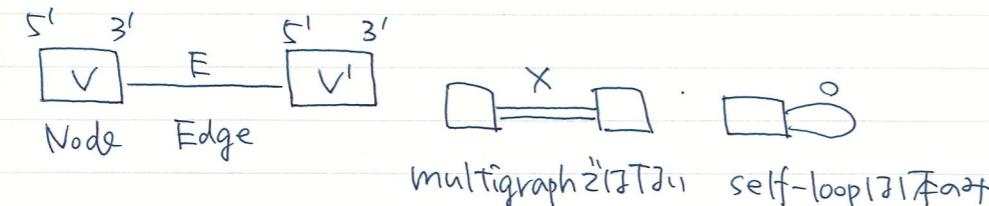


Graph Burrows-Wheeler Transform (gPBWT)

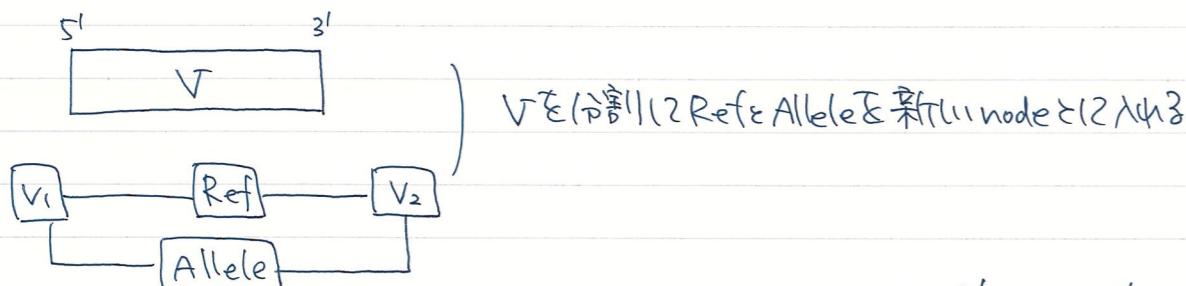
PBWT では各 site の binary が完全に順序が逆転している必要がある。
 → データグラフ上に thread は 1 ハロタイプを埋め込むこと。ハロタイプ全体の圧縮、検索を容易
 (→ 2. 理論上は diploid × 100000 も圧縮可能)

Definition

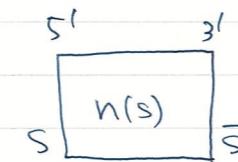
- $G = (V, E)$: 双方向グラフ



- 重複の削除: short indels / SNVs を埋め込みの上簡単な V を分割する。



- 側面 s の反対側の側面を \bar{s} , s' 属する node と $n(s)$ と呼ぶ



- 方向をもつない配列(これは undirected graph と同型)を Ambisequence と呼ぶ。

- どう G 上に haplotype (オーバルの集合) & thread と呼ぶか。

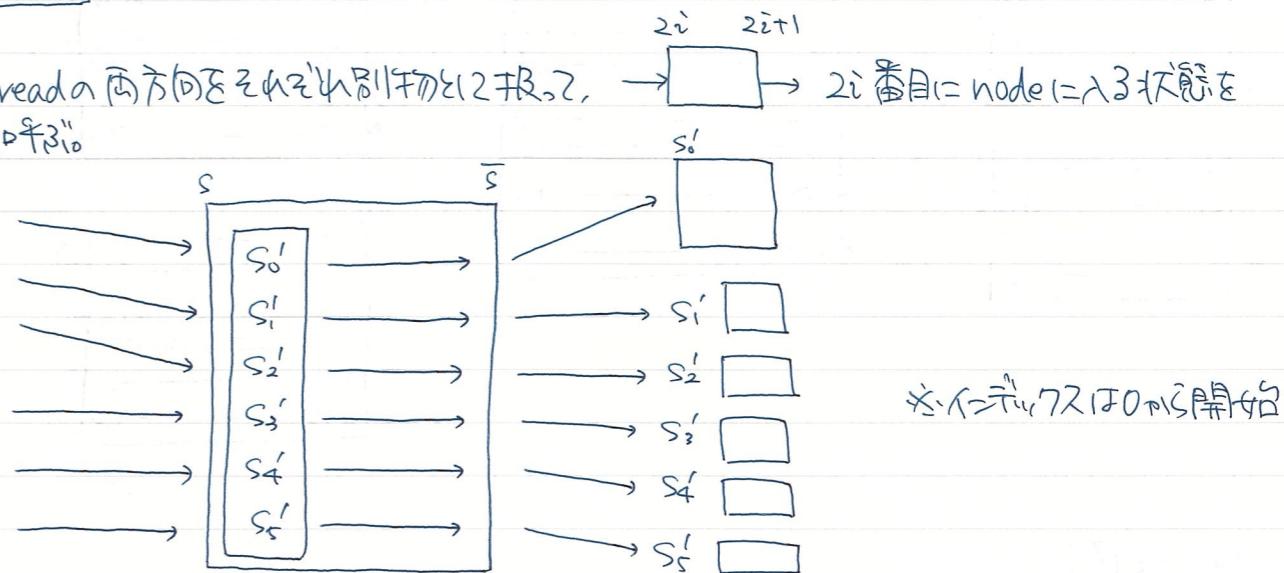
$T = [T_0, T_1, \dots, T_{2N-2}, T_{2N-1}]$ で, $0 \leq i < N$ は T_i と T_{2i+1} が, どう上の半定の 1 ハロの両側面に
 対応すると仮定する。Thread もオーバル ambisequence である。

- $G = \text{haplotypes} \& \text{threads}$ の集合と埋め込み, これを T とおく。

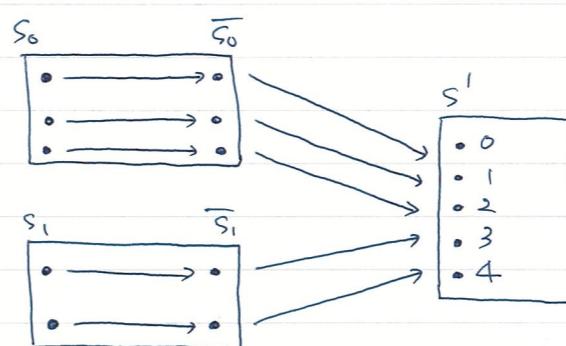
- G と T は $T = \{T_i\}$ に保存する, 任意の配列と一致する haplotypes を T の中からどのみに
 検索する, と呼ぶ。

Graph PBWT

各 thread の両方向をそれぞれ別物とし T_i と \bar{T}_i , $\bar{T}_i \rightarrow 2i$ 番目 node (= 入力状態) を
 「visits」と呼ぶ。



側面 s = 並びで s の 2 つの thread & reversed sort (2. s と \bar{s} の側面 s' の値を並べる。
 例: $s = 12345$ reversed sort は 12435 で $B_s[]$ で実現)。



$c(\bar{s}, s')$

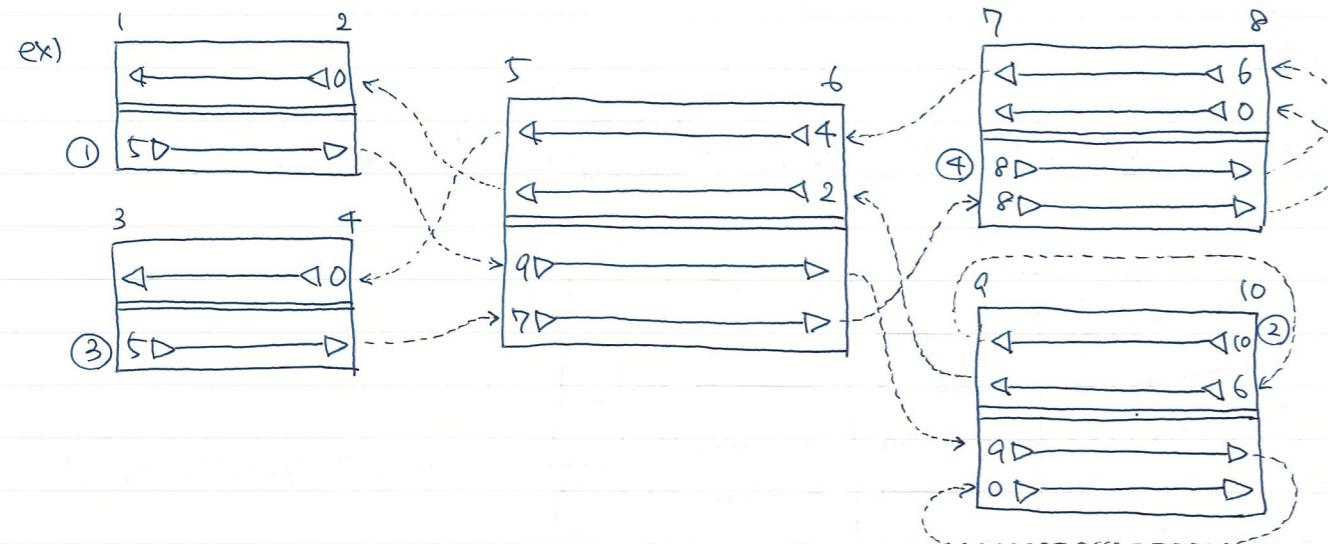
$B_s[]$ の中で, $\bar{s} \rightarrow s'$ の edge を通じて移動 (2 つ = threads が最初に現れる位置を示す)。

この定義で, s' は s の edge を持つ, 2 つの側面
 \bar{s}, s ($\bar{s} < s$) は $\bar{s} > s$ 。

$$c(\bar{s}_0, s') = 0, c(\bar{s}_1, s') = 3$$

$c(\bar{s}_0, s') \leq c(\bar{s}_1, s')$

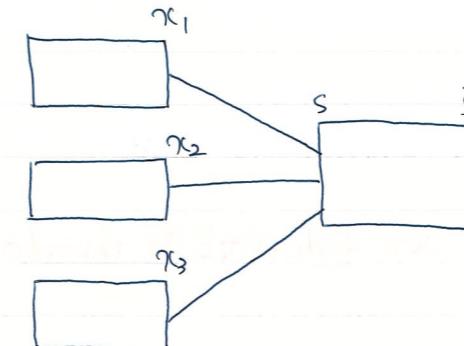
どう G と threads の集合 $T = \{T_i\}$ は $c(\cdot)$ 関数と $B[\cdot]$ の配列を組み合わせて T と gPBWT と呼ぶ。



side	$B_s[]$	$C()$
1	5	
2	0	$C(5,2)=0$
3	5	
4	0	$C(5,4)=0$
5	9,7	$C(2,5)=0, C(4,5)=1$
6	4,2	$C(7,6)=0, C(9,6)=1$
7	8,8	$C(6,7)=1$
8	6,0	$C(8,8)=0$ も初めに出現するイニテル、クスと書く
9	9,0	$C(6,9)=0, C(0,9)=1$
10	0,6	$C(9,10)=1$

* ここで $c(\bar{s}_0, s') \leq c(\bar{s}_1, s')$ は成り立つ

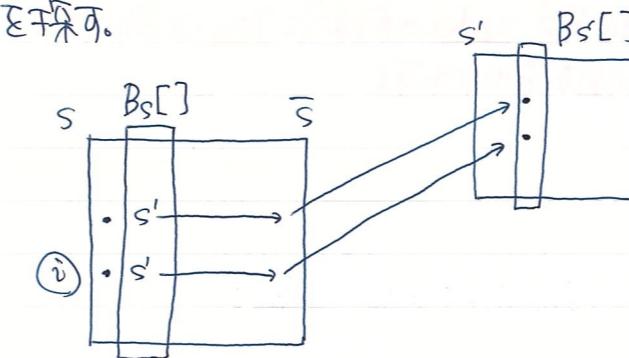
Algorithm 1 $G, B[], C()$ が与えられた時に、任意の thread をどの辺に復元するか?



$C(x_1, s), C(x_2, s), \dots$ の中の最小値より、 s' は必ず thread の数に一致する。これは b とす。

(s は流入辺 ≤ 3 edge の「T」の「T」。 $B_s[]$ の長さが ≤ 3 である。 s' は始まる thread の数に一致する。)

まず、ab1回の thread の開始を特定し、次に $B_s[i]^*$ 。これは $[0, b]$ の重みを $= 2$ 。 $n(s)$ の = 1 node を示す。



次に B_s^* におけるイニテル、クス

$C(\bar{s}, s') +$ $\lceil B_s[i]^* \rceil = B_s[i]^*$ の値が ≤ 2 あるものにいくつあるか?

!!

$\lceil B_s[] \rceil \oplus \lceil B_s[i]^* \rceil$ は何番目の s' か? - 1

次に、 $B_s[]$ の該当のイニテル、クスを貯め、 s', \bar{s}' の次の node に \oplus する。

XF = ハセキル T2, $B_s[]$ のこの値 $\oplus 0 (= T)$ と $\oplus 2$ で \oplus して T の thread を復元する。

この作業量は、thread の数 M 、各々の thread の長さ N とすると、側面から「そこまで始めた M thread を探す」という作業を、すべての側面に H を \oplus 行う必要がある。全体の作業量は $O(MNT)$ は T と

Code

Algorithm 1 G と $gPBWT$ を与え threads を取る。 ≤ 3

```
function Starting-at (Side, G, B[], C())
  if Side has incident edges then:
    return c(s, Side) for minimum s over all "sides" connected to Side.
  else:
    return length (BSide[ ])
```

(イニシエーティング・オブ・サイド) [スレッド数] = 2. 二つのサイドに2つのスレッドを割り当てる。

```
function Rank(b[], Index, Item):
  Rank ← 0
  for all i in b[] (0 ≤ i < Index) do:
    if b[i] = Item then Rank ← Rank + 1
  return Rank
```

b[] の中で Index の前までは Item と同じ値を示す個数をカウントする

```
function Where-to (Side, Index, B[])
  return c (Side, BSide[Index]) + Rank(BSide[ ], Index, BSide[Index])
  △ Index がどこで元におい何番目か。
```

```
function Extract (G, C[], B[])
  for all side S in G do:
    TotalStarting ← Starting-at (s, G, B[], C())
    for all i in [0, TotalStarting) do:
      Side ← s, Index ← i, Orientation ← [S, S], NextSide ← BSide[Index]
      while NextSide ≠ 0 do:
        Orientation ← Orientation + [NextSide, NextSide]
        Index ← Where-to (Side, Index, B[], C())
        Side ← NextSide
        NextSide ← BSide[Index]
      yield Orientation
```

スレッドの通り水と格納する

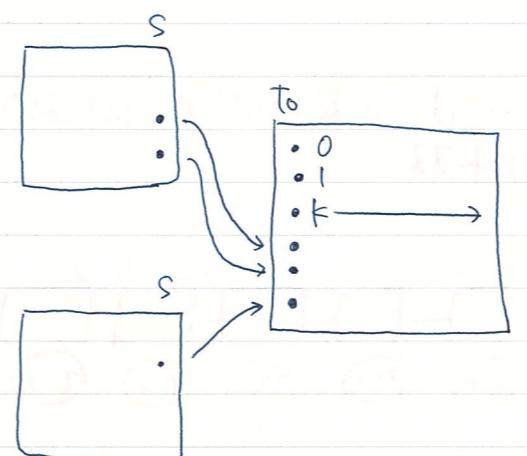
この操作は Rank 関数に以下の rank query (3の値がリスト内に何番目にいるか?) の計算が必須である。bit vectors と wavelet tree の相性が良い。

100GP などは 1 本自体は巨大なグラフあるが、threads 数は 1 つ T と $B[]$ を 1 つまとめて保存するには妥当である。

Algorithm 2 おいて構築エラー $gPBWT$ は、新しい thread をどう追加するか

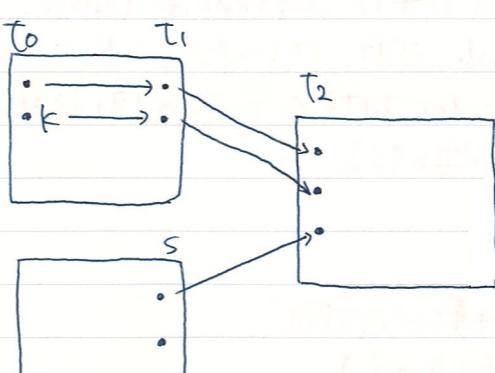
(new thread は 2 方向をもつ)、そのための片方 $T_{new} = [t_0, t_1, \dots, t_{2N}, t_{2N+1}]$ (≥ 12) とする。

最初の定義だと t_{2N-2}, t_{2N-1} がおれ! $T = T_0 \cup T_1 \cup \dots \cup T_{2N-2}$ が本文もこうしている



- ① $T_{new}[j]$ $BTo[]$ の最初の値 = λ 。
(Algorithm 2 では幾何的に最初 = λ にする)

$T_{new}[j]$ $BTo[]$ の k 番目 (= λ) と β と。 $BTo[k] = t_2$.



- ② $C(s, t_0) \rightarrow T_{new}$ $X = t_0, t_1, \dots, t_{2N-1}, t_{2N}$.

- ③ t_2 中で $t_1 \rightarrow t_2$ edge を通じて 3 番目の頂点 = β T_{new} の順番を数え、 β と $C(t_1, t_2)$ の値を T_{new} $BTo[]$ に追加 $T = T_0 \cup T_1 \cup \dots \cup T_{2N-2} \cup T_2$, T_2 を求める。(Where-to を使う)

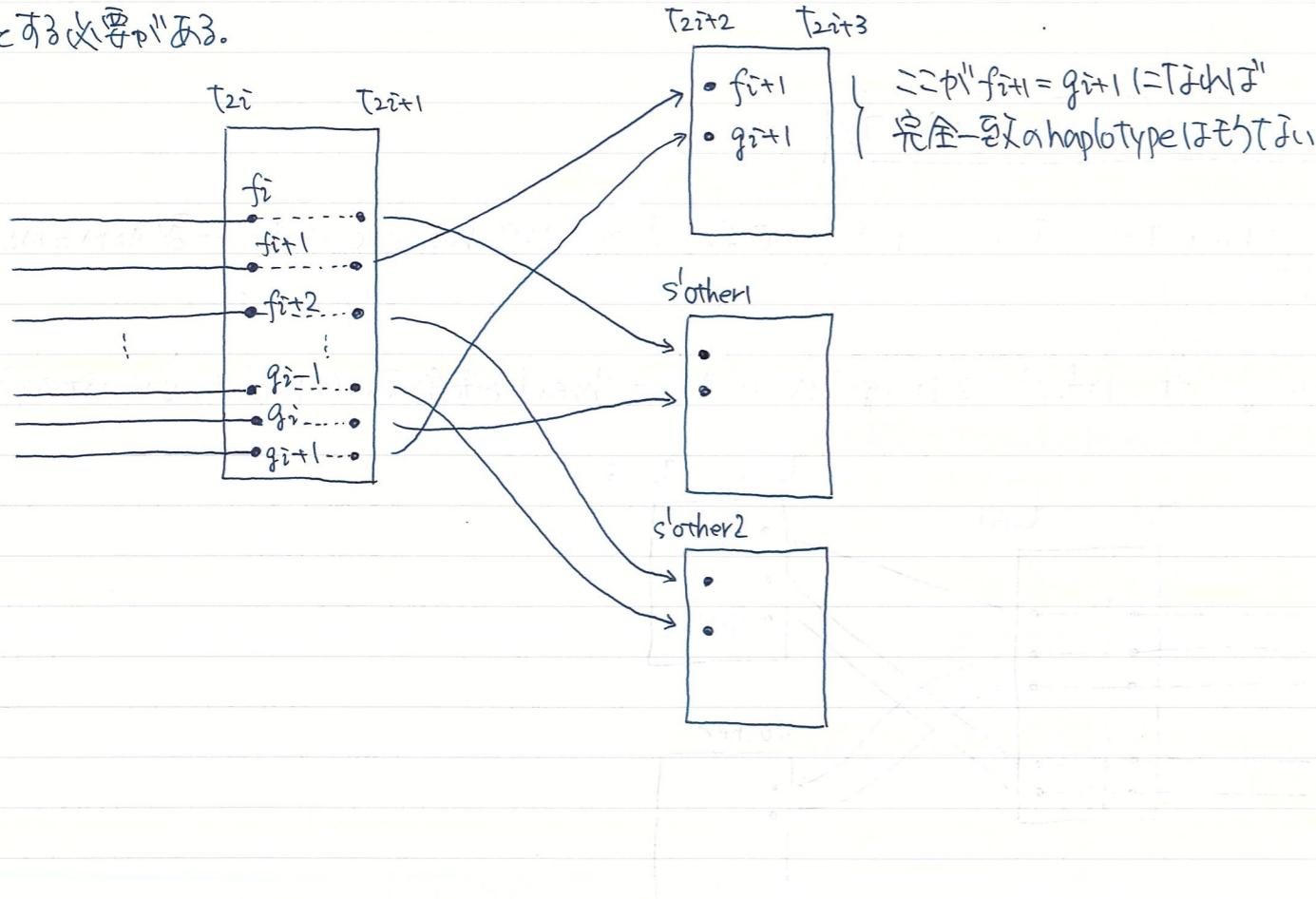
- ④ $s \rightarrow T, T$ に t_2 を追加 $C(s, t_2)$ を β に追加する。

$T = T^{\text{hap}}$, (本文中には書いた通り) f_i と g_i の haplotype $^{(1)}$, そして T_2i+2 が行くルートは全て無い。

つまり, (*) の where T_0 は 実装とは違うので不適である。(*の通り)

| f_i を含むそれ以後の $T_2i \rightarrow T_2i+1 \rightarrow T_2i+2$ と行くもののがない, 2つを f_{i+1}
| g_i を含むそれ以後の $T_2i \rightarrow T_2i+1 \rightarrow T_2i+2$ と行くものがない, 2つを g_{i+1}

とする必要がある。



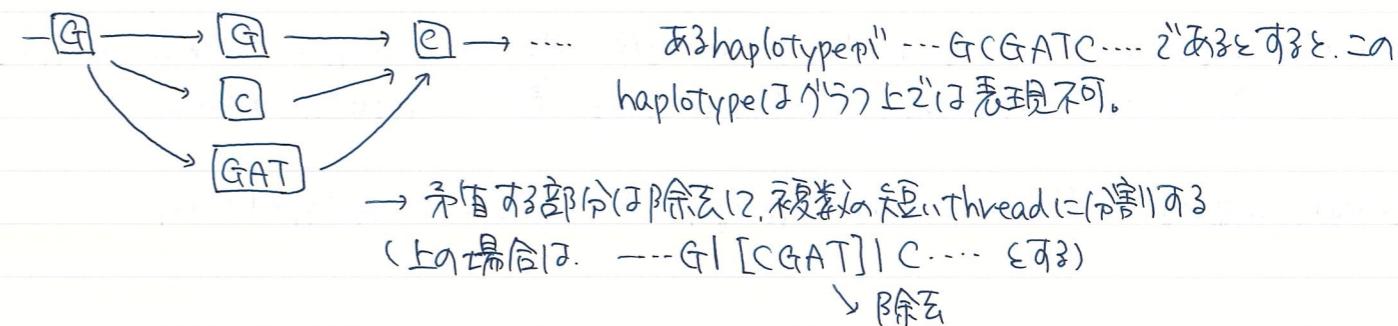
Results (結果)

- qPBWT (q. vg toolkit の component) で 3×8 の 23 個の haplotype

→ Chr22 に 23 個の "haplotype" を vg を用いて作成し, これは VCF から得られる 5008 haplotypes の情報を埋め込んでいる。qPBWT 対応 Xg は T^{hap} , 2つとも各納入。

(構造と qPBWT の形式が各納入)

- $T = T^{\text{hap}}$, いくつも haplotype があるが、上位 2 つが表現されてるのか?



- 2504 samples の "最初から最後まで" diploid を維持するにはどうすればいい? 部分的に haploid でも可いの?

- Algorithm 3 を用いて haplotype の埋め込みを行うと (毎 2 つ haplotype と一度言及する) 278GB² の 9h19mn. 最終的な Xg は T^{hap} , サイズは [436MB]。

haplotype

- "haplotype" 100bpの haplotype を 100万本 simulation 生成し、各々の haplotype と完全一致する haplotype の数を 確定する。
- 全体の 46% (J) の haplotype と match できる。完全一致する haplotype の数は本文 Fig.3 (基本的には 1 本あたり 1 hit となる) と共通性が高い (高さが hit 1 本の場合には整合する haplotype の多くなる)

Read alignment

- NAI2878 (1000G Low-coverage reads を用い) 2. Xg/GCSA2-based mapper 2. chr 22 に mapping.
→ リードを "スコア" で選別 (2. primary / secondary mapping は 整合する haplotype を求める)。
- primary mapping の 0.17%, secondary mapping の 0.73% は 整合する haplotype に含まれる。