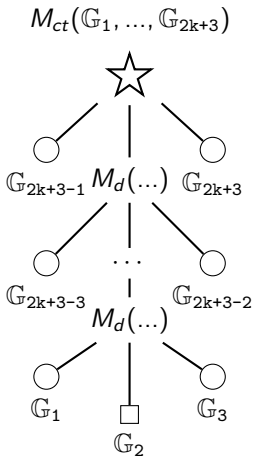
Algorithms in Comparative Genomics

Universität Bielefeld, WS 2020/2021
Dr. Marília D. V. Braga · Leonard Bohnenkämper

https://gi.cebitec.uni-bielefeld.de/teaching/2020winter/cg

Exercise sheet for the holidays, 17.12.2020

Exercise 1 (Christmas Tree Median) (10* pts)



Given $2k+3$ canonical genomes $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, ...\mathbb{G}_{2k+3}$ and an algorithm to compute the median $M_d(\mathbb{A}, \mathbb{B}, \mathbb{C})$ of three genomes $(\mathbb{A}, \mathbb{B}, \mathbb{C})$ under a distance model d. The *Christmas Tree Median*[1] is defined as

$$M_{ct}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, ...\mathbb{G}_{2k+3-2}, \mathbb{G}_{2k+3-1}, \mathbb{G}_{2k+3}) = M_d(\mathbb{G}_{2k+3-1}, M_{ct}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, ...\mathbb{G}_{2k+3-2}), \mathbb{G}_{2k+3}) \quad (1)$$

with recursion base

$$M_{ct}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3) = M_d(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3) \quad (2)$$

1. Compute the Christmas Tree Median of the following genomes under the breakpoint distance ($d = d_{BP}$): $\mathbb{G}_1 = (\ 1\ 2\ 3\ 4\ )$, $\mathbb{G}_2 = (\ \bar{2}\ \bar{1}\ \bar{4}\ \bar{3}\ )$, $\mathbb{G}_3 = (\ 2\ 3\ 4\ 1\ )$, $\mathbb{G}_4 = [\ 1\ \bar{3}\ \bar{2}\ 4\ ]$, $\mathbb{G}_5 = [\ \bar{2}\ \bar{1}\ \bar{4}\ \bar{3}\ ]$
2. Disprove (e.g. via counter example): The Christmas Tree Median under the breakpoint distance is always a breakpoint median.
3. Given the order of the genomes may not be permuted, is the Christmas Tree Median under the SCJ distance ($d = d_{SCJ}$) unique? Argue why/why not (Spoiler).
4. Prove or disprove: No metric d on a set with two or more distinct elements exists, under which the Christmas Tree Median is always a true Median[2] (Spoiler 1, 2, 3, 4, 5).

Exercise 2 (Double Distances) (8* pts)

Regard the genomes $\mathbb{S} = (\ 1\ 2\ 3\ 4\ 5\ )$ and $\mathbb{D} = (\ \bar{1}\ 5\ \bar{5}\ 1\ 2\ \bar{4}\ \bar{3}\ 2\ 3\ 4\ )$.

1. Calculate the breakpoint double distance $d^2_{BP}(\mathbb{S}, \mathbb{D}) = d_{BP}(\mathbb{S} \oplus \mathbb{S}, \mathbb{D})$ and give an optimal matching $\mathbb{M}_{\mathtt{opt}}$ on $\mathbb{S} \oplus \mathbb{S}$ and $\mathbb{D}$.
2. Calculate the SCJ double distance $d^2_{SCJ}(\mathbb{S}, \mathbb{D})$ between $\mathbb{S}$ and $\mathbb{D}$.
3. Is the optimal matching for the breakpoint double distance $\mathbb{M}_{\mathtt{opt}}$ also optimal under the SCJ distance? Can you generalize your observation?

[1] which I made up; don't look for this in the literature ;)
[2] The true median of a set $K \subseteq S$ under metric d on space S being the element $M_d \in S$ that minimizes $\sum_{k \in K} d(M_d, k)$.

4. Computing the DCJ double distance is NP-hard. Using the matching from subtask 1 what is the DCJ distance $d_{DCJ}(\mathbb{M}_{\text{opt}}, \mathbb{D})$ between $\mathbb{M}_{\text{opt}}$ and $\mathbb{D}$?

5. Find another matching $\tilde{\mathbb{M}}_{\text{opt}}$ on $\mathbb{S} \oplus \mathbb{S}$, which minimizes $d_{SCJ}(\tilde{\mathbb{M}}_{\text{opt}}, \mathbb{G})$, but produces a different DCJ distance from the one computed in subtask 4, i.e. $d_{DCJ}(\tilde{\mathbb{M}}_{\text{opt}}, \mathbb{G}) \neq d_{DCJ}(\mathbb{M}_{\text{opt}}, \mathbb{G})$.

Exercise 3 (Fun with Programming) (6*pts)

Let's do something a bit different! Implement the following tasks in a language of your choice. Maybe use a language you don't see every day - the more obscure, the better (though please dont hand in `brainfuck` code :)

1. Read two unichromosomal linear genomes as lists of signed integers from the command line (either as a parameter or during runtime) and output the type of genome pair (canonical, singular, balanced, natural). For example:

```
>./myprogram -g1 1 -3 -2 4 5 -g2 1 2 3 4 5
>This is a canonical genome pair (i.e. natural, singular and balanced).
```

2. If the genomes are canonical, calculate their SCJ-distance and display it via the command line.

3. Calculate an optimal SCJ-sorting scenario sorting the first genome into the second and display it by writing out the current chromosomes and where the next cut/join is applied in each step; i.e.

```
[1 | -3 -2 4 5]
[1] [-3 -2 | 4 5]
[1 *] [-3 -2 *] [4 5]
[1 2 3 *] [* 4 5]
[1 2 3 4 5]
```

Exercise 4 (Extending DCJ) (3* pts)

Let $d_{DCJ/Inv/SCJ}(\mathbb{G}_1, \mathbb{G}_2)$ be the minimum operations to transform canonical genomes $\mathbb{G}_1$, $\mathbb{G}_2$ into each other, where each operation may be an SCJ- or DCJ-operation or an inversion.

1. Show that
$$d_{DCJ/Inv/SCJ}(\mathbb{G}_1, \mathbb{G}_2) = d_{DCJ}(\mathbb{G}_1, \mathbb{G}_2) \tag{3}$$
(Spoiler)

2. What might be disadvantages of having such a general model as DCJ?

Exercise 5 (Santa's Unsigned Inversion Distance) (8* pts)

A snowstorm has caused chaos in Santa's workshop! The n presents which are usually nicely ordered from $1, ..., n$ are now in a random permutation $r_1, .., r_n$. Fortunately the elves can use magic to invert any segment $r_i, r_{i+1}, ...r_{i+k-1}, r_{i+k}$ to $r_{i+k}, r_{i+k-1}, ..., r_{i+1}, r_i$ in constant time.

1. Sort the following pile of presents twice using
   (a) signed inversions (as discussed in the lecture)
   (b) unsigned inversions (as the elves use)

   1 4 2 3 5.

2. Give a short description of how and why a signed inversion sorting scenario can be mapped to an unsigned one and why it is therefore possible to sort the presents in $\mathcal{O}(n)$ time.

3. After a while you notice that the elves seem to be doing a lot of unnecessary inversions. You start to suspect that sorting unsigned permutations that are not already sorted with unsigned inversions can always be done in fewer steps than with signed inversions. Why might that be?

4. The algorithm you described in subtask 2 sorts a list of length n in $\mathcal{O}(n)$ time. Why doesn't this conflict with the the theoretical bound of $\Omega(n log(n))$ you are familiar with for sorting? (Spoiler)

☆☆☆ **Have fun, stay safe and enjoy your holidays!** ☆☆☆