heidikarkkainen@Heidi-MacBook-Air 03-permutation % java -Xms12g -Xmx12g -jar target/02-permutation-1.0-SNAPSHOT-jar-with-dependencies.jar

NOTE: Running this app will take __a long time__ , be patient...

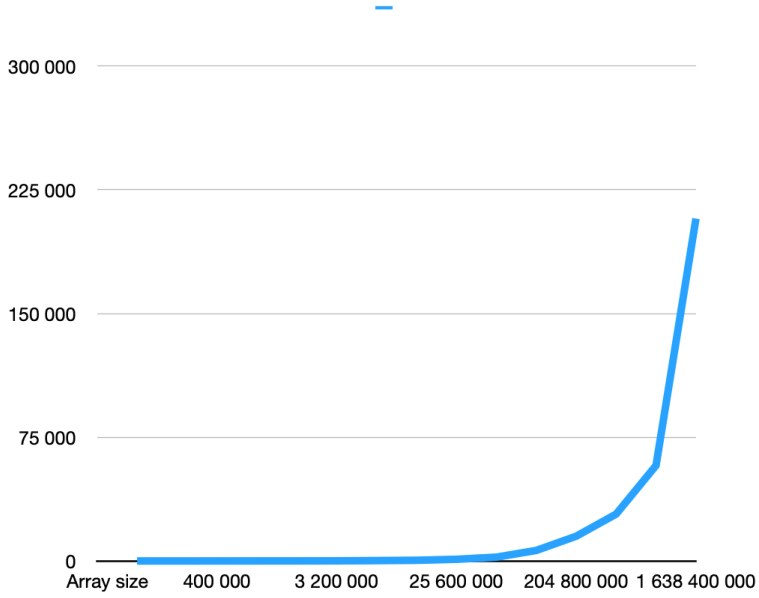
OS: Mac OS X 10.16 - x86_64 - processors: 4 - JVM heap size: 12 884 901 888

Array size	ms
100 000	14
200 000	23
400 000	28
800 000	54
1 600 000	59
3 200 000	104
6 400 000	260
12 800 000	448
25 600 000	1 051
51 200 000	2 348
102 400 000	6 417
204 800 000	15 036
409 600 000	28 433
819 200 000	57 911
1 638 400 000	207 538

Integer overflow, Java only allows int (signed 32bit) as array index.

Trying out with 2 000 000 000...

Array size	ms
100 000	14
200 000	23
400 000	28
800 000	54
1 600 000	59
3 200 000	104
6 400 000	260
12 800 000	448
25 600 000	1 051
51 200 000	2 348
102 400 000	6 417
204 800 000	15 036
409 600 000	28 433
819 200 000	57 911
1 638 400 000	207 538



Luulen, että aikakompleksisuus on neliöllinen eli O(nˆ2), koska suurilla taulukoilla taulukon koon kaksinkertaistuessa aika enemmän kuin kaksinkertaistuu. Toinen vaihtoehto, jota mietin, on O(n) eli lineaarinen. Algoritmissä on kaksi for-silmukkaa, jotka eivät ole sisäkkäin. Luentomateriaalin mukaan tällaisen koodin aikakompleksisuus on O(n). Koodin perustella aikakompleksisuus siis olisi lineaarinen mutta suoritusajan perusteella neliöllinen.

Taulukon kokoa voisi kasvattaa käyttämällä long-tietotyyppiä int-tietotyypin sijasta, jolloin voitaisiin viitata useampaan taulukon alkioon. Jos tietotyypiksi vaihdetaan long, käy kuitenkin niin, että tietokoneen muisti loppuu kesken ennen kuin voidaan saavuttaa niin suuri taulukko kuin mikä on mahdollista long-tietotyypillä.