Project1

Algorithm

This algorithm uses a binary tree T for comparison and an array A to store all the indexes of numbers.

- (i) Init T as a **13-layer binary tree** and T has 3334 leaves. Init A as a **10000-size array**. Divide A into 3334 groups, in which the first 3333 groups contain 3 numbers in each and the last group has only one number.
- (ii) In each group, find the largest number then put the max to the first position in that group. Put all the max numbers into T's leaves.
- (ii') We can use an array Flag to store the whether the numbers in a group can be sorted after the comparison of step (ii). If Z is the largest nubmer in group (X_i, Y_i, Z_i), then after the comparison of step (ii), the order among X_i, Y_i, Z_i could be known. Then we record the order of X_i and Y_i in Flag[i] for future use. **(optimization we made)**
- (iii) Compare each two leaves which have the same parent, then put the larger leaf's value into parent. Iterate until reaches T's root and we find the largest number among all the leaves. Add root's value to Best array.
- (iv) Then we adjust A and T. Assume now the root's value is v, then set $A[\lfloor \frac{v-1}{3} \rfloor \times 3] = -1$. If all the numbers in this group is -1, then assign the leaf corresponding to the group to -1. If there is only one number of this group is not -1, assign the leaf corresponding to the group to the value of this number. If two numbers of this group are not -1, assign the leaf corresponding to the group to the max value of this two numbers and swap the first one of the group and the max value of this two numbers.

- (v) Now we only need to compare this node and its neighbor child and update their parent until reaches the root (If -1 occurs, ignore the node). Add root's value to Best array.
- (vi) Repeat step (iv)(v) until find k largest numbers.

Analaysis

1. Theoretical WC

- The process of building the tree:
  - The array A is made of triples. There are $\lfloor \frac{n}{3} \rfloor = 3333$ triples and one node left. For each group, we need 2 **COMPARE** to get the largest number of this group. That is $3333 \times 2 = 6666$.
  - T is a binary tree, every two nodes need 1 **COMPARE** to get the largest number of them. That's $1667 + 833 + 417 + 208 + 104 + 52 + 26 + 13 + 7 + 3 + 2 + 1 = 3333$.
  - So we use $6666 + 3333 = 9999$ COMPARE for building the tree.
- The process of fetching 40 largest number's from the tree:
  - Adjust the array A. We at most need 1 **COMPARE** if two of the triple is not -1 and our optimization doesn't come into play.
  - Update the tree T. We need 1 **COMPARE** in every layer besides the last one. And we have 13 layers. So that is 12.
  - We need to fetch 40 largest number's from the tree, so we only need 39 Array Adjust and 39 Tree Update.

    That's $39 \times (12 + 1) = 507$.

So the theoretical WC is $9999 + 507 = 10506$.

2. Theoretical expected WC

- The process of building the tree:
  - The array A is made of triples. There are $\lfloor \frac{n}{3} \rfloor = 3333$ triples and one node left. For each group, we need 2 **COMPARE** to get the largest number of this group. That is $3333 \times 2 = 6666$.
  - T is a binary tree, every two nodes need 1 **COMPARE** to get the largest number of them. That's

$1667 + 833 + 417 + 208 + 104 + 52 + 26 + 13 + 7 + 3 + 2 + 1 = 3333$
.
    - So we use $6666 + 3333 = 9999$ COMPARE for building the tree.
- The process of fetching 40 largest number's from the tree:
    - Adjust the array A. We at most need 1 **COMPARE** if two of the triple is not -1 and our optimization doesn't come into play. We run 1000 times and we need 39 Array Adjust each time. So that's

      $39 \times (1 - (1 - (\frac{2}{3})^{39})^{1000}) + \dots$

      For clear, we set $F(k) = 1 - (1 - C_{39}^{k}(\frac{1}{3})^{k}(\frac{2}{3})^{39-k})^{1000}$ when k >= 0 and 0 when k=-1.

      Then we got $\sum_{k=0}^{38}(39-k) \times [\Pi_{k=-1}^{k-1}(1 - F(k-1))] \times F(k) \approx 32.49$
    - Update the tree T. We need 1 **COMPARE** in every layer besides the last one. And we have 13 layers. So that is 12.
    - We need to fetch 40 largest number's from the tree, so we only need 39 Array Adjust and 39 Tree Update.

      That's $39 \times 12 + 32.49 = 500.49$.

So the theoretical WC is $9999 + 507 = 10499.49$.

3. Theoretical AVG

- The process of building the tree:
    - The array A is made of triples. There are $\lfloor \frac{n}{3} \rfloor = 3333$ triples and one node left. For each group, we need 2 **COMPARE** to get the largest number of this group. That is $3333 \times 2 = 6666$.
    - T is a binary tree, every two nodes need 1 **COMPARE** to get the largest number of them. That's
      $1667 + 833 + 417 + 208 + 104 + 52 + 26 + 13 + 7 + 3 + 2 + 1 = 3333$
      .
    - So we use $6666 + 3333 = 9999$ COMPARE for building the tree.
- The process of fetching 40 largest number's from the tree:
    - Adjust the array A. The case that Z is the largest nubmer in group (X_i, Y_i, Z_i) happens by the probability 1/3. And the probility that we visit the same group during the first 39 fetches is very small comparing to 1/3. So we roughly need $\frac{2}{3}$ **COMPARE**.

- Update the tree T. We need 1 **COMPARE** in every layer besides the last one. And we have 13 layers. So that is 12.
- We need to fetch 40 largest number's from the tree, so we only need 39 Array Adjust and 39 Tree Update.

  That's $39 \times (12 + 2/3) = 494$.

So the theoretical WC is $9999 + 494 = 10493$

4. Observed WC and AVG and explain inconsistencies

```
n=10000,  k=40:  maximum= 10499,  avg=10489.49
```

Observed WC is 10499 and Observed AVG is 10489.49.

The inconsistencies between Observed WC and Theoretical expected WC: ***1000 is not large enough***

The inconsistencies between Observed AVG and Theoretical AVG: ***1000 is not large enough and we might visit the same group during the first 39 fetches***.