ALGO Programming Assignment1

by 108062138 Po-Yu,Wu

hackmd https://hackmd.io/@sBeNJ4fqRNqa67PhyWWV4A/BkdbJFgUK

(https://hackmd.io/@sBeNJ4fqRNqa67PhyWWV4A/BkdbJFgUK)

source code https://github.com/108062138/MedOfMedSel.git

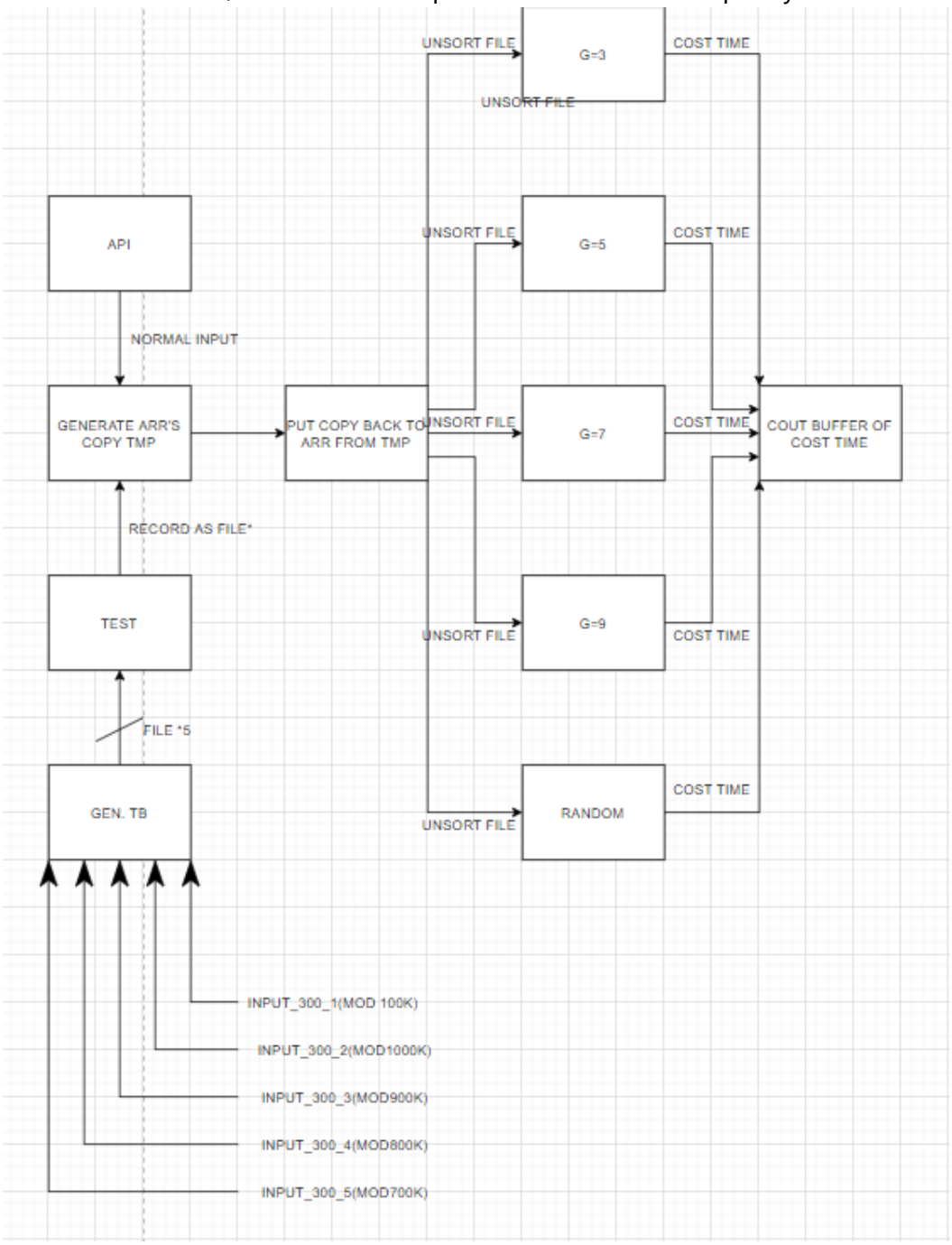
(https://github.com/108062138/MedOfMedSel.git)

1. Your report must contain the flowchart or the pseudo code of your program. You have to describe how your approach works.

> flow chart: I will draw two flow charts: `main` chart and `KTH` chart First chart is about main's code.

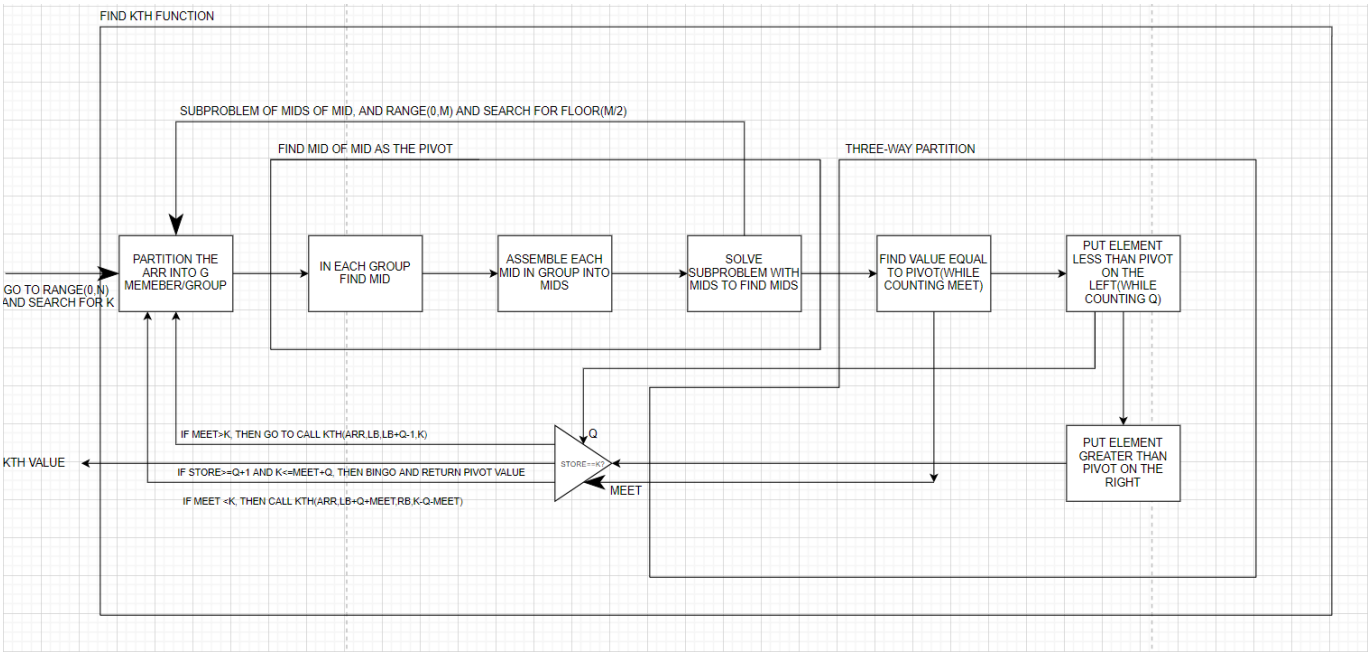
- For `main` chart:The rectangle with G=? represent G, which means the call of `KTH` with ? as the group size. The cout buffer enables me to monitor each testcase's practical running time. One thing needs to be noticed is that ARR requires to be re-filled with its duplicated array, denoted as TMP, because ARR will be partial sorted after calling function `KTH` . In the

submitted version, this redundant part is removed for simplicity.



- For `KTH` chart: This chart is basically composed of three components: `find mid of mid as pivot`, `pick mid of mid as pivot and partition the array`, `deter next recursion`.
  - `find mid of mid as pivot` takes three procedures:
    - Splict the original array into several size-G array
    - Among each size-G array pick the mid point(Since G must be odd number, the mid point must always exist)
    - Assemble those mid points into a new array, denoted as `mids`, and call `KTH` to solve the problem: “find the mid of mids”
  - `pick mid of mid as pivot and partition the array`: takes five procedures:
    - Find the first index s.t arr[index] is same as the mid of mids pivot
    - Swap that with the end of the array

- PARTITION the rest of the array just as in quick sort's PARTITION. While partitioning the array, we can get q(number of elements that are smalle than the pivot) and meet (number of elements that value are same as pivot) in passing. (this part can be optimized)
- Decide next recursion step based on store
  - if q==k, then bingo and return the value(this part can be optimized into k>=q+1 && k<=q+meet)
  - if q>k, this means left recursion will be selected. So call `KTH(arr,lb,lb+q-1,k)`
  - if q<k, this means right recursion will be selected. So call `KTH(arr,lb+q+meet,rb,k-q-meet)`



2. You should compare the running time of your algorithm with the input elements are divided into groups 3, 5, 7, 9, and Randomized-Select. Average the execution timeof 20 experiments for each group size and Randomized-Select. Besides, you must analyze the time complexity in different group sizes and show your results.

- time complexity of G=3:
  - Consider the worst case: $n-2(\lceil\lceil n/3\rceil/2\rceil-2)=2n/3+4$ for the recursion part to call KTH ,n for the partition part, and $\lceil n/3\rceil$ for find mid of mid by call KTH . The time complexity $T(n)=T(\lceil n/3\rceil)+an+T(2n/3+4)$
  - Assume that T(n)=O(n) works in cases $n_0 \dots n-1$. That is $T(n)\leq cn/3+c+2cn/3+4c+an=cn+5c+an\geq cn$ The contradiction happens. Thus, T(n)'s upper bound is great than O(n), which implies that at G=3, T(n) is not linear.

- time complexity of G=5:
  - Consider the worst case: $n-3(\lceil\lceil n/5\rceil/2\rceil-2)=7n/10+6$ for the recursion part to call KTH ,n for the partition part, and $\lceil n/5\rceil$ for find mid of mid by call KTH . The time complexity $T(n)=T(\lceil n/5\rceil)+an+T(7n/10+6)$
  - Assume that T(n)=O(n) works in cases $n_0 \dots n-1$. That is $T(n)\leq cn/5+c+7cn/10+6c+an=9cn/10+7c+an=cn+(-cn/10+7c+an)\leq c$ Let $(-cn/10+7c+an)\leq 0$, and we have $10an/(n-70)\leq c$. Selecting $n_0=140$ with $c=20a$ will make case n work. Thus, $T(n)=O(n)$.

- time complexity of G=7:
  - Consider the worst case: $n-4(\lceil\lceil n/7\rceil/2\rceil-2)=5n/7+8$ for the recursion part to call KTH ,n for the partition part, and $\lceil n/7\rceil$ for find mid of mid by call KTH . The time complexity $T(n)=T(\lceil n/7\rceil)+an+T(5n/7+8)$
  - Assume that $T(n)=T(\lceil n/7\rceil)+an+T(5n/7+8)$ works in cases $n_0 \dots n-1$. That is $T(n)\leq cn/7+c+cn5/7+8c+an=cn+(-cn/7+9c+an)\leq cn$ Let $(-cn/7+9c+an)\leq 0$, and we have $7an/(n-63)\leq c$. Selecting $n_0=126$ with $c=14a$ will make case n work. Thus, $T(n)=O(n)$.

- time complexity of G=9:
  - Consider the worst case: $n-5(\lceil\lceil n/9\rceil/2\rceil-2)=13n/18+10$ for the recursion part to call KTH ,n for the partition part, and $\lceil n/9\rceil$ for find mid of mid by call KTH . The time complexity $T(n)=T(\lceil n/9\rceil)+an+T(13n/18+10)$
  - Assume that $T(n)=T(\lceil n/9\rceil)+an+T(13n/18+10)$ works in cases $n_0 \dots n-1$. That is $T(n)\leq cn/9+c+13cn/18+10c+an=cn+(-cn3/18+11c+an)\leq cn$ Let $(-cn3/18+11c+an)\leq 0$, and we have $6an/(n-66)\leq c$. Selecting $n_0=132$ with $c=12a$ will make case n work. Thus, $T(n)=O(n)$.

- I use 4 different test bench, each contains 3000k input,range(0,1000k). The result is written in the following table.

unit:ms

N=30000000 kth=10000000

trail of input_300_1	G=3	G=5	G=7	G=9	random
1	129007	15279	12067	11184	2149
2	134738	15856	12401	11562	2074
3	120454	14723	14237	11301	2060
4	134977	16901	12844	11683	2433
5	131437	17470	12847	12131	2077

> N=30000000 kth=10000000

trail of input_300_2	G=3	G=5	G=7	G=9	random
1	125268	15268	12430	11523	2177
2	123266	15804	11915	11079	2160
3	118178	15357	12084	11100	2079
4	131591	15797	12161	11393	2188
5	125998	13293	10192	9939	3117

> N=30000000 kth=2000000

trail of input_300_3	G=3	G=5	G=7	G=9	random
1	194637	16147	13061	10546	729
2	246261	20482	14892	13269	965
3	231268	19539	13628	12497	1143
4	221733	16583	12148	12088	812
5	206582	16724	12483	11166	674

> N=30000000 kth=20000000

trail of input_300_4	G=3	G=5	G=7	G=9	random
1	140855	15359	11961	10750	2892
2	150143	16289	12569	11347	3078
3	151269	16596	12466	11606	3160
4	147922	16215	12270	11559	3168
5	146582	16224	12317	10425	3032

The average time for G=3 at N=30000k is roughly 149558.3ms
The average time for G=5 at N=30000k is roughly 16295.3ms
The average time for G=7 at N=30000k is roughly 12548.6ms
The average time for G=9 at N=30000k is roughly 10863.4ms
The average time for random at N=3000k is roughly 2108.3ms

It seems that the for every test case, the random selection almost guarantees the best performance. On top of that, random selection is approximately 100x faster than median of median selection.

As for those median of median selections, G=3 always has the worst running time(the reason is given above: G=3 will not have a linear complexity s.t. G=3 takes the most time).

Discard G=3, the remaining median of median selections are G=5, G=7, G=9. In my computer, there is a trend that the running time gradually decreases at the ratio of 0.8x along as the increment of G .

But even the best case amonh the median of median median of median selection, which is G=9 in my laptop, is way slower than random selection.

3. Optimization

> These following three optimizating methods are all offered by 108062101張智孝.

- use german flag method to make elements, which have same value as the pivot, to assemble together. This procedure enables the following optimization.

```
int q=0;
int meet = 0;
int l = lb-arr.begin(); int r = rb-arr.begin();
int i = lb-arr.begin();
while(i<=r){
    if (arr[i] < midValue) {
        swap(arr[l], arr[i]);
        q++;
        l++;
    }
    else if (arr[i] > midValue) {
        swap(arr[i], arr[r]);
        r--;
        i--;
    }
    else meet++;
    i++;
}
```

- augument the possibility to early return by making k be quite easy to activate early return. The reason why I can directly rewrite `k==q+1` into `k>=q+1 && k<=q+meet` is because I meet will guarantee to be greater than

```
//if(k==q+1) return midValue;<-old approach
if(k>=q+1 && k<=q+meet) return midValue;<-new approach
if (k <= q)
    return KTH(arr,lb,lb+q-1, k);
else
    return KTH(arr,lb+q+meet,rb, k - (q + meet));
```

- pass iterator, aka smart pointer , and reference instead of copyign the entire vector

```
int KTH(vector<int> &arr,vector<int>::iterator lb, vector<int>::iterator rb,int k){
...
```

> TO BE HONESST, WITHOUT THE HELP FROM 108062101張智孝, THE PROJECT WILL NOT PROCEED IN TIME...