Mohammad Arshad Hossain Ratul

1930319

① Merge sort, though it is not in-place and has a time complexity of $O(n)$ but we have extra enough memory to sort. As quick sort's worst case can be $O(n^2)$ and works with small dataset so we can say quick sort is not applicable here and as merge sort has ~~time~~ worst time complexity $O(n \log n)$ so it is suitable.

i	
1	
2	
4	
8	

②
```
while (n>=0)
{
    for (i=1; i<logn; i=i×2)   — n+1
    {
        p++;    — loglogn
    }
    if(n<100)
    {
        k++    — ~~100~~ O(1)
    }
    else    — O(1)
    { m++
    }
    n=n-1
}
```

$T(n) = (n+1)(\log\log n) + c + c$

$= n\log\log n + \log\log n + c + c$

$\therefore O(n\log\log n)$

3 Heap sort, as heap sort is in-place ~~and~~, capable of working with large dataset ~~and~~ and duplicate values and also has a time complexity of O(nlogn) hence Heap sort is suitable here.

(4) Char a[ ] = { } ← given

```
int n = a.size().
Char b[n];
Quick sort(a);    ← nlogn

int h = n;

for(i=0 ; i<n ; i++)  — n
{
   if i is even
   {   b[i] = a[h]
       h--;
   }
   else
   {
       b[i] = a[i]
   }
```

Check_palindrome( ): — n

$T(n) = n\log n + n + n + c$

$= O(n\log n)$

Scanned with CamScanner

(5) No, As linked list itself has a complexity of $O(n)$. Therefore if we use linked list the overall complexity of heap becomes $O(n^2 \log n)$ which is much worse ~~than~~ than any available algorithm.

(7)

(6) No it is not possible. As we all know topological sort only deals with directed acyclic graph (DAG). Therefore even though the negative cycles removed positive cycles are present, in order to do topological sort we must not have any cycles.

~~(7) I will use Bellman-Ford, as di. Dijkstra always dont give correct answer for negative path.~~

(7) I will use Dijkstra algorithm for this specific graph as it will not ~~eateut~~ relax visited nodes.
I will not use Bellman-ford as it wont be able to relax at V-1; time.

SA	AB	AC	BD	CD
0	∞	∞	∞	∞
	-9	-9	-7	-7
	-8	-8	-6	-6
	-7	-7	-5	-5
	-6	-6	-4	-4
	-5	-5	-3	-3

(Prove)