Tirgul 7

Heap Sort

some more stuff

Build Heap

- We have an array of numbers which we want to turn into a heap

Approach 1:

```
for i=1,...,n {
 insert (A[i])
}
```

(due to convexity)

(worst case) Complexity: $\sum_{i=1}^{n} \log i \geq \sum_{i=n/2}^{n} \log \frac{n}{2} = n(\log n - \log 2) = \Omega(n \log n)$

(will happen for some input, for example a sorted array)

Approach 2:

```
for i=⌊n/2⌋ down to 1 {
 max_heapify(A[i])
}
```


Build Heap

- Complexity:

- We have a trivial upper bound of $n \cdot \log(n)$

- In fact, we can achieve a tighter bound:

$$T(n) = 1\tfrac{n+1}{2} + 2\tfrac{n+1}{4} + 3\tfrac{n+1}{8} + \ldots + \log n = \sum_{k=1}^{\log n} k \tfrac{n+1}{2^k} = (n+1)\sum_{k=1}^{\log n} \tfrac{k}{2^k}$$

$$= (n+1)\begin{bmatrix} \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + \tfrac{1}{n} \\ + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + \tfrac{1}{n} \\ + \tfrac{1}{8} + \ldots + \tfrac{1}{n} \\ \ldots \\ + \tfrac{1}{n} \end{bmatrix} = (n+1)\sum_{i=0}^{\log n} \tfrac{1}{2^i} \leq 2(n+1) = O(n)$$

Heap Sort

```
HEAPSORT(A)
1 BUILD-MAX-HEAP(A)
2 for i ← length[A] downto 2
3     do exchange A[1] ↔ A[i]
4        heap-size[A] ← heap-size[A] - 1
5        MAX-HEAPIFY(A, 1)
```

Complexity:

- line 1 (build heap): $O(n)$
- lines 2-5 $n \cdot \log(n)$

➲ altogether $O(n \cdot \log(n))$

Sorting - Summary

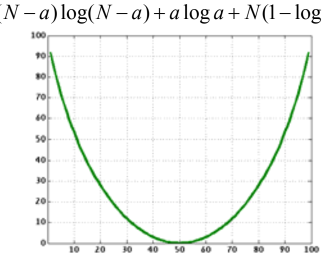
Algorithm	Average	Worst	Randomized	Additional Memory	Stable ?	Data types
Radix sort	n	n	-	O(k+n)	yes	integers
Bobble sort	n^2	n^2	-	O(1)	yes	comparable
Insertion sort	n^2	n^2	-	O(1)	yes	comparable
Merge sort	$n \cdot \log(n)$	$n \cdot \log(n)$	-	O(n)	yes	comparable
Heap sort	$n \cdot \log(n)$	$n \cdot \log(n)$	-	O(1)	no	comparable
Quick sort	$n \cdot \log(n)$	n^2	$n \cdot \log(n)$	O(1)	no	comparable

Ex4 – Q2.a

We need to show that:

$(N-a)\log(N-a) + a\log a + N \geq N \log N$

or that $(N-a)\log(N-a) + a\log a + N(1-\log N) \geq 0$



Ex4 – Q2.a

$$f(x) = (N-a)\log(N-a) + a\log a + N(1-\log N)$$
$$f'(x) = -\log(N-a) - 1 + \log a + 1 = \log a - \log(N-a)$$
$$f'(x) = 0 \Rightarrow a = N - a \Rightarrow 2a = N$$

We still have to decide whether the point is a local maxima, minima or an inflection point, but we know that:

$$f(0) = N$$
$$f(N) = N$$
$$f(N/2) = 0$$

➲ $N/2$ is a local minima → $f(x) \geq 0$

Ex4 – Q2.b

- We have shown that the average depth of leaves in a tree containing n leaves is $\log(n)$
- When we fix the random decisions made by a random sorting algorithm we get a decision tree
- The number of leaves in this tree is $n!$ (although the algorithm is random, we still need to be able to reach all the possible permutations)
- We need to show that there is at least one input, for which the average of all routes from root to leaf in all of the random decision trees is $n\log(n)$ (remember, when we want to compute random complexity, we calculate an average of all the random decisions)

Ex4 – Q2.b

- We can number the permutations from 1 to $n!$
- A correct sorting of an input is a permutation in which $a_i < a_j$ for each $i < j$
- We know that the average depth of the leaves in each of these trees is $n\log(n)$
- There is at least one permutation, p, which matches a leaf that has an average depth of at least $n\log(n)$ in all of the trees (remember that each leaf is a possible permutation of the input)

➲ The random complexity of the algorithm (any algorithm) is $\Omega(n\log(n))$