**CS 120: Intro to Algorithms and their Limitations**

Week 0

Pset Due: September 13, 2023 **Denny Cao**

---

## §1 Google Search Algorithm

1. Calculate Page Rank $PR(\text{url})$ for each url. $PR(\text{url}) \in [0, 1]$
2. Given search term $w$, find set $S_w$ of webpages that contain $w$ (They can cache this info and store page ranks for popular search terms $\rightarrow$ less computation)
3. Return list of urls in $S_w$ in *descending* order of $PR(\text{url})$

## §2 The Sorting Problem

**Input:** An array $A$ of pairs of key-value pairs $((k_0, v_0), (k_1, v_1), ..., (k_{n-1}, v_{n-1}))$

- Key is PR and value is url

**Output:** An array $A'$ of key-value pairs sorted $((k'_0, v'_0), (k'_1, v'_1), ..., (k'_{n-1}, v'_{n-1}))$

**Definition 2.1** (Sorted)**.** Conditions needed to be sorted:

1. Sorted by keys, i.e: $k'_0 \le k'_1 \le ... \le k'_{n-1}$
2. $\exists$ permutation $\pi : [n] \rightarrow [n]$ st $(k'_i, v'_i) = (k_{\pi(i)}, v_{\pi(i)}) \forall i$

   **Notation 2.2** (Subset of Natural Numbers) As they are used quite often, we denote $[n]$ to mean the natural numbers up to $n$.

Currently in descending order. Set $k$ to be $1 - PR$ and it will be in ascending order which is what we want

**Issue:** If there are duplicate keys, then it is not unique. Sorting algorithm output can return them in any order and still satisfy these conditions.

### §2.1 Exhaustive-Search Sort (ESS)

For each permutation $\pi : [n] \rightarrow [n]$: if $k_{\pi(0)} \le k_{\pi(1)} \le ... \le k_{\pi(n-1)}$ then: return $((k_{\pi(0)}, v_{\pi(0)}), (k_{\pi(1)}, v_{\pi(1)}), ..., (k_{\pi(n-1)}, v_{\pi(n-1)}))$

- Check all permutations and return when reach a permutation where it is in ascending order

**Will it always return a correct answer?**

- Goal of Correctness Proof: For every input, algorithm gives a valid output to complete the computational problem.

1. For every input, $\exists$ valid output
2. If $\exists$ a valid output, ESS returns something

3. If ESS returns something, it returns something correct

*Proof of Correctness of ESS.* We will not prove 1 as it is trivial. For 2, we cannot get past correct output without returning something. For 3, we only return when we checked if keys are in order and thus the return is the correct output. □

## §2.2 Insertion Sort

For $i \in [n]$ : put $A[i]$ into correct place (less than the following element and greater than or equal to preceding element) in $[A[0], ..., A[i-1]]$ return $A$

*Proof of Correctness for Insertion Sort.* Prove by induction on $i$. Let $P(i)$ be the statement that, at the end of loop step $i$ of the loop, $[A[0], A[1], ..., A[i]]$ is a correct sorting of first $i+1$ elements.

*Base Case*: $i = 0$, $[A[0]]$ is a valid sort.

*Inductive Step*: Insert $A[i+1] = (k, v)$ into correct place in $[A[0], ..., A[i]]$ which are correctly sorted by induction.

*After Insertion*:

- Inserting element doesn't change set of elements, ie. can modify permutation by inserting $i+1$
- $k[0] \leq k[1] \leq ... \leq k[j] \leq k[j+1] \leq ... \leq k[i]$. By induction, it is in order except for $i$, and by inserting into the correct location, remains correct order.

$P(n)$ is the statement "when insertion sort finishes, it's correct." □

- Note that here, $A[0]$ is being used to refer to the returned array. A more precise way is to use $A_i[k]$, which is the $k$th element at the $i$th iteration.