HW9: Algorithms

Due: Dec 3, 2024 @ 11:59 PM

Problem 1 [10 pts]: Log Practice

1. $x = \log_2 64$:

$$\begin{aligned}\log_2 64 &= x\\ 2^x &= 64.\end{aligned}$$

   Since $2^6 = 64$, $x = 6$.

2. $5 = \log_2 x$:

$$\begin{aligned}\log_2 x &= 5\\ 2^5 &= x.\end{aligned}$$

   Therefore, $x = 32$.

3. $3 = \log_x 8$:

$$\begin{aligned}\log_x 8 &= 3\\ x^3 &= 8.\end{aligned}$$

   Since $2^3 = 8$, $x = 2$.

4. $x = \log_2\left(\frac{1024}{17}\right) + \log_2 17$:

$$\begin{aligned}x &= \log_2\left(\frac{1024}{17}\right) + \log_2 17\\ &= \log_2 1024 - \log_2 17 + \log_2 17\\ &= \log_2 1024.\end{aligned}$$

   Since $1024 = 2^{10}$, $x = 10$.

5. $x = \log_2 1024^{17}$:

$$\begin{aligned}x &= 17 \cdot \log_2 1024\\ &= 17 \cdot 10\\ &= 170.\end{aligned}$$

Problem 2 [20 pts]: Sorting Steps

Given List: $18, 45, 23, 2, -5, 10, 99, 0$

(i) Insertion Sort

$$\begin{aligned}
&\text{Start:} && \square\, 18, 45, 23, 2, -5, 10, 99, 0\\
&\text{Step 1:} && 18\square\, 45, 23, 2, -5, 10, 99, 0\\
&\text{Step 2:} && 18, 45\square\, 23, 2, -5, 10, 99, 0\\
&\text{Step 3:} && 18, 23, 45\square\, 2, -5, 10, 99, 0\\
&\text{Step 4:} && 2, 18, 23, 45\square\, -5, 10, 99, 0\\
&\text{Step 5:} && -5, 2, 18, 23, 45\square\, 10, 99, 0\\
&\text{Step 6:} && -5, 2, 10, 18, 23, 45\square\, 99, 0\\
&\text{Step 7:} && -5, 2, 10, 18, 23, 45, 99\square\, 0\\
&\text{Final:} && -5, 0, 2, 10, 18, 23, 45, 99.
\end{aligned}$$

(ii) Merge Sort

$$\begin{aligned}
&\text{Step 1: } [18, 45, 23, 2] \quad \text{and} \quad [-5, 10, 99, 0]\\
&\text{Step 2: } [18, 45], [23, 2] \quad \text{and} \quad [-5, 10], [99, 0]\\
&\text{Step 3: } [18], [45], [23], [2], [-5], [10], [99], [0]\\
&\text{Merge: } [18, 45], [2, 23], [-5, 10], [0, 99]\\
&\text{Merge: } [2, 18, 23, 45], [-5, 0, 10, 99]\\
&\text{Final: } [-5, 0, 2, 10, 18, 23, 45, 99].
\end{aligned}$$

Problem 3 [24 pts]: Three Stooges

We determine the smallest n (whole number) that ensures faster sorting for each case.

1. **Larry's computer sorts faster than Moe's:**

   Larry's sorting time:
   $$T_L(n) = 2\sqrt{n},$$
   Moe's sorting time:
   $$T_M(n) = n.$$
   For Larry's computer to sort faster than Moe's:
   $$T_L(n) < T_M(n),$$
   $$2\sqrt{n} < n.$$
   Dividing through by $\sqrt{n}$ (valid for $n > 0$):
   $$2 < \sqrt{n}.$$
   Squaring both sides:
   $$n > 4.$$
   Answer: The smallest n such that Larry's computer sorts faster than Moe's is:
   $$n = 4.$$

2. **Curly's computer always sorts faster than Moe's:**

   Curly's sorting time:
   $$T_C(n) = \log_2 n,$$
   Moe's sorting time:
   $$T_M(n) = n.$$
   Explanation: For all $n > 0$, $\log_2 n$ grows slower than n. Therefore:
   $$T_C(n) < T_M(n) \quad \text{for all } n > 0.$$
   Answer: Curly's computer always sorts faster than Moe's for $n > 0$.

3. **Curly's computer always sorts faster than Larry's:**

   Curly's sorting time:
   $$T_C(n) = \log_2 n,$$
   Larry's sorting time:
   $$T_L(n) = 2\sqrt{n}.$$
   Explanation: For all $n > 0$, $\log_2 n$ grows slower than $2\sqrt{n}$. Therefore:
   $$T_C(n) < T_L(n) \quad \text{for all } n > 0.$$
   Answer: Curly's computer always sorts faster than Larry's for $n > 0$.

Problem 4 [24 pts]: Recurrence

We are solving the recurrence relation:

$$T(n) = T(n-2) \cdot 7,$$

with the base case $T(1) = 1$.

Solution

Step 1: Unroll the recurrence relation
Starting from $T(n) = T(n-2) \cdot 7$:

$$\begin{aligned} T(n) &= T(n-2) \cdot 7 \\ &= \big(T(n-4) \cdot 7\big) \cdot 7 \\ &= T(n-4) \cdot 7^2. \end{aligned}$$

Substituting $T(n-4)$:

$$\begin{aligned} T(n) &= \big(T(n-6) \cdot 7^2\big) \cdot 7 \\ &= T(n-6) \cdot 7^3. \end{aligned}$$

Generalizing for k steps:

$$T(n) = T(n-2k) \cdot 7^k.$$

—

Step 2: Determine the stopping condition
The recurrence stops when $n - 2k = 1$ because $T(1)$ is the base case. Solving for k:

$$\begin{aligned} n - 2k &= 1 \\ k &= \frac{n-1}{2}. \end{aligned}$$

—

Step 3: Plug $k = \frac{n-1}{2}$ into the general form
Substitute $k = \frac{n-1}{2}$ into $T(n) = T(n-2k) \cdot 7^k$:

$$T(n) = T(1) \cdot 7^{\frac{n-1}{2}}.$$

Since $T(1) = 1$, this simplifies to:

$$T(n) = 7^{\frac{n-1}{2}}.$$

—

Final Answer

$$T(n) = 7^{\frac{n-1}{2}}$$

Problem 5 [2 extra credit pts]: Bubble Sort Extra Credit

(i) Bubble Sort's Advantage

Bubble Sort has an advantage over other sorting methods for nearly-d-sorted lists because it performs localized swaps, ensuring that each element moves closer to its sorted position in every pass. Unlike other sorting methods, Bubble Sort avoids unnecessary comparisons and shifts for elements already near their correct positions. This localized behavior minimizes unnecessary operations, making Bubble Sort particularly efficient for nearly-sorted lists.

(ii) Why Bubble Sort Needs Only d Passes

Bubble Sort works by repeatedly passing through the list, swapping adjacent elements that are out of order. For a nearly-d-sorted list:

1. Each element is at most d positions away from its sorted position.
2. During a single pass, any element can move at most one position closer to its correct position because Bubble Sort swaps adjacent elements.
3. Therefore, in the worst case, it takes at most d passes to move every element into its correct position.

Explanation: An element x that is d positions away from its correct position will require at most d swaps to "bubble" to its correct position. The algorithm terminates early when no swaps are needed, optimizing its performance for nearly-sorted lists.

Example

Consider the nearly-d-sorted list $A = [1, 5, 10, 15, 9, 20, 34, 57, 91, 66]$ ($d = 2$):

- In the first pass, the out-of-order element 9 moves from index 4 to its correct position at index 2, and 66 moves from index 9 closer to index 8.
- In the second pass, 66 reaches its correct position at index 8.
- After $d = 2$ passes, the list is fully sorted.

Thus, Bubble Sort takes at most d passes to sort a nearly-d-sorted list.