هر سوال را در محل در نظر گرفته شده پاسخ دهید. پاسخ های خارج از محل تصحیح نمیشوند. نام و شماره دانشجویی را روی تمام برگه ها بنویسید. شماره دانشجویی باید با اعداد لاتین نوشته شود. مهلت این تمرین شنبه ۲۰ مهر ماه است.

پریسا یلسوار ـ سید صالح اعتمادی

۱. [۳۴] هر گروه از توابع زیر را به ترتیب افزایش نرخ رشد مرتب کنید.

(آ)

$$f_1(n) = n^{0.99999}\log n \qquad f_2(n) = 1000000n \qquad f_3(n) = 1.0000001n \qquad f_4(n) = n^2$$

$$f_1(n) < f_3(n) < f_2(n) < f_4(n)$$

To see why $f_1(n)$ grows asymptotically slower than $f_2(n)$, recall that for any $c > 0$, $\log n$ is $\mathcal{O}(n)$. Therefore we have:

$$f_1(n) = n^{0.99999}\log n = \mathcal{O}(n^{0.99999}.n^{0.000001}) = \mathcal{O}(n) = \mathcal{O}(f_2(n))$$

The function $f_2(n)$ is linear, while the function $f_4(n)$ is quadratic, so $f_2(n)$ is $\mathcal{O}(f_4(n))$. Finally, we know that $f_3(n)$ is also linear, which grows slower than quadratic and $f_2(n)$, so $f_3(n)$ is also $\mathcal{O}(f_4(n))$.

(ب)

$$f_1(n) = 2^{2^{1000000}} \qquad f_2(n) = 2^{100000n} \qquad f_3(n) = \binom{n}{2} \qquad f_4(n) = n\sqrt{n}$$

$$f_1(n) < f_4(n) < f_3(n) < f_2(n)$$

The variable n never appears in the formula for $f_1(n)$, so despite the multiple exponentials, $f_1(n)$ is constant. Hence, it is asymptotically smaller than $f_4(n)$, which does grow with n. We may rewrite the formula for $f_4(n)$ to be $f_4(n) = n\sqrt{n} = n^{1.5}$. The value of $f_3(n)$ is given by the formula $n(n-1)/2$, which is $\Theta(n^2)$. Hence, $f_4(n) = n^{1.5} = O(n^2) = O(f_3(n))$. Finally, $f_2(n)$ is exponential, while $f_3(n)$ is quadratic, meaning that $f_3(n)$ is $\mathcal{O}(f_2(n))$.

(ج)

$$f_1(n) = n^{\sqrt{n}} \qquad f_2(n) = 2^n \qquad f_3(n) = n^{10}.2^{\frac{n}{2}} \qquad f_4(n) = \sum_{i=1}^{n}(i+1)$$

$$f_4(n) < f_1(n) < f_3(n) < f_2(n)$$

To see why, we first use the rules of arithmetic series to derive a simpler formula for $f_4(n)$:

$$f_4(n) = \sum_{i=1}^{n}(i+1) = \frac{n((n+1)+2)}{2} = \frac{n(n+3)}{2} = \Theta(n^2)$$

This is clearly asymptotically smaller than $f_1(n) = n^{\sqrt{n}}$. Next, we want to compare $f_1(n)$, $f_2(n)$, and $f_3(n)$. To do so, we transform both $f_1(n)$ and $f_3(n)$ so that they look more like $f_3(n)$:

$$f_1(n) = n^{\sqrt{n}} = (2^{\log n})^{\sqrt{n}} = 2^{\sqrt{n}.\log n}$$
$$f_3(n) = n^{10}.2^{\frac{n}{2}} = 2^{\log n^{10}}.2^{\frac{n}{2}} = 2^{\frac{n}{2}+10\log n}$$

The exponent of the 2 in $f_1(n)$ is a function that grows more slowly than linear time; the exponent of the 2 in $f_3(n)$ is a function that grows linearly with n. Therefore, $f_1(n) = \mathcal{O}(f_3(n))$. Finally, we wish to compare $f_3(n)$ with $f_2(n)$. Both have a linear function of n in their exponent, so it's tempting to say that they behave the same asymptotically, but they do not. If c is any constant and $g(x)$ is a function, then $2^{cg(x)} = (2^c)^{g(x)}$. Hence, changing the constant of the function in the exponent is the same as changing the base of the exponent, which does affect the asymptotic running time. Hence, $f_3(n)$ is $\mathcal{O}(f_2(n))$, but $f_2(n)$ is not $\mathcal{O}(f_3(n))$.

۲. [۱۶] پیچدگی محاسباتی هر یک از روابط بازگشتی زیر را برای $T(n,n)$ بنویسید.

(a)

$$T(x,c) = \Theta(x) \quad \text{for } c \leq 2,$$
$$T(c,y) = \Theta(y) \quad \text{for } c \leq 2 \text{ , and}$$
$$T(x,y) = \Theta(x+y) + T(x/2, y/2)$$

The correct answer is $\Theta(n)$. To see why, we rewrite the recurrence relation to avoid Θ notation as follows:

$$T(x,y) = c(x+y) + T(x/2, y/2)$$

We may then begin to replace T(x/2, y/2) with the recursive formula containing it:

$$T(x,y) = c(x+y) + c(\frac{x+y}{2}) + c(\frac{x+y}{4}) + c(\frac{x+y}{8}) + \ldots$$

This geometric sequence is bounded from above by $2c(x+y)$, and is obviously bounded from below by $c(x+y)$. Therefore, $T(x,y)$ is $\Theta(x+y)$, and so $T(n,n)$ is $\Theta(n)$.

(b)

$$T(x,c) = \Theta(x) \quad \text{for } c \leq 2,$$
$$T(c,y) = \Theta(y) \quad \text{for } c \leq 2 \text{ , and}$$
$$T(x,y) = \Theta(x) + T(x, y/2)$$

The correct answer is $\Theta(n \log n)$. To see why, we rewrite the recurrence relation to avoid Θ notation as follows: $T(x,y) = cx + T(x, y/2)$. We may then begin to replace $T(x, y/2)$ with the recursive formula containing it: $T(x,y) = \underbrace{cx + cx + cx + \cdots + cx}_{\Theta(\log y) times}$. As a result, $T(x,y)$ is $\Theta(x \log y)$. When we substitute n for x and y, we get that $T(n,n)$ is $\Theta(n \log n)$.

(c)

$$T(x,c) = \Theta(x) \quad \text{for } c \leq 2,$$
$$T(x,y) = \Theta(x) + S(x, y/2),$$
$$S(c,y) = \Theta(y) \quad \text{for } c \leq 2,$$
$$S(x,y) = \Theta(y) + T(x/2, y)$$

The correct answer here is $\Theta(n)$. To see why, we want to first eliminate the mutually recursive recurrence relations. To do so, we will replace all references to the function $S(x,y)$ with the definition of $S(x,y)$. This yields the following recurrence relation for $T(x,y)$:

$$T(x,y) = \Theta(x) + \Theta(y/2) + T(x/2, y/2)$$

We can rewrite this to eliminate the constants and get the recurrence $T(x,y) = \Theta(x+y) + T(x/2, y/2)$. This is precisely the same recurrence relation as seen in part (a) of this problem, so it must have the same complexity.

۳. [۲۰] درستی یا نادرستی عبارات زیر را با علامت (✓) یا (✗) مشخص کنید. دلیل خود را در نقطه چین زیر هر عبارت توضیح دهید.

(a) ✗ A $\Theta(n^2)$ algorithm always takes longer to run than a $\Theta(\log n)$ algorithm.

False. The constant of the $\Theta(\log n)$ algorithm could be a lot higher than the constant of the $\Theta(n^2)$ algorithm, so for small n, the $\Theta(\log n)$ algorithm could take longer to run.

(b) ✓ If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $h(n) = \Theta(f(n))$.

True. Θ is transitive

(c) ✓ If $f(n) = O(g(n))$ and $g(n) = \mathcal{O}(h(n))$, then $h(n) = \Omega(f(n))$

True. $\mathcal{O}$ is transitive, and $h(n) = \Omega(f(n))$ is the same as $f(n) = \mathcal{O}(h(n))$.

(d) ✗ If $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(f(n))$ then $f(n) = g(n)$.

False: $f(n) = n$ and $g(n) = n + 1$.

۴. [۳۰] آرایه A با n عنصر صحیح و عدد صحیح x به شما داده شده است. الگوریتمی با پیچگی زمانی $\Theta(n\log n)$ طراحی کنید که تعیین کند آیا دو عنصر در این آرایه با مجموع x وجود دارد یا خیر.(دو عنصر الزاما متمایز نیستند.)

We first sort the elements in the array using a sorting algorithm such as merge-sort which runs in time $\Theta(n \log n)$. Then, we can find if two elements exist in A whose sum is x as follows. For each element $A[i]$ in A, set $y = A[i]-x$. Using binary search, find if the element y exists in A. If so, return $A[i]$ and y. If we can't find y for any $A[i]$, then return that no such pair of elements exists. Each binary search takes time $\Theta(\log n)$, and there are n of them. So, the total time for this procedure is $T(n) = \Theta(n \log n)+\Theta(n \log n)$ where the first term comes from sorting and the second term comes from performing binary search for each of the n elements. Therefore, the total running time is $T(n) = \Theta(n \log n)$. An alternate procedure to find the two elements in the sorted array is given below:

SUM-TO-X(A)

```
Merge-Sort(A)
i ← 1
j ← length(A)
while i ≤ j do
    if A[i] + A[j]  equals  x then
        return A[i], A[j]
    end
    if A[i] + A[j]  <  x then
        i ← i + 1
    end
    if A[i] + A[j]  >  x then
        j ← j − 1
    end
end
```

We set counters at the two ends of the array. If their sum is x, we return those values. If the sum is less than x, we need a bigger sum so we increment the bottom counter. If the sum is greater than x, we decrement the top counter. The loop does $\Theta(n)$ iterations, since at each iteration we either increment i or decrement j so ji is always decreasing and we terminate when $ji < 0$. However, this still runs in time $\Theta(n \log n)$ since the running time is dominated by sorting.