A0149963M.

Assignment 1 : YSC 2229 E 2.

Date No.

(2) procedure. bitcount (S: bit string) | Cost. | iterations.

count := 0. | c_1 | 1

while $S \neq 0$. | c_2 | n.

count := count + 1. | c_3 | n

$S := S \wedge (S-1)$. | c_4 | $n \times n = n^2$

return. count [count is no. of 1s in S]. | c_5 | 1

$$\therefore T(n) = c_1 + n(c_2 + c_3) + n^2(c_4) + c_5$$

Reason why $S := S \wedge (S-1)$ is that to create the new S, one must at worst case scan the S and (S-1) n times. In addition, (S-1)'s creation might at worst case take O(n) too.

(3) Guess: $\boxed{T(n) = O(n^2)}$.

(4) for some positive constant a,

$$c_5 + c_1 + n(c_2 + c_3) + n^2(c_4) \leq an^2.$$

Let $c_6 = c_2 + c_3$ and

$c_7 = c_1 + c_5$.

$$c_7 + nc_6 + n^2(c_4) \leq an^2$$

$$an^2 - c_4 n^2 - c_6 n - c_7 \geq 0.$$

Let $\boxed{b = a - c_4}$, $\boxed{d = c_6 + c_7}$.

$$bn^2 \geq c_6 n + c_7.$$

$$bn^2 \geq (c_6 + c_7)n \geq c_6 n + c_7.$$

$bn^2 \geq dn$: TRUE if

$n \geq 1$ and $b > d$.

$$a - c_4 \geq c_6 + c_7$$

$$a \geq c_4 + c_6 + c_7$$

$$= c_4 + c_1 + c_2 + c_3 + c_5.$$

(Main). assume n is length of list poly. Date cost No. iteration

2(1) procedure compute_polynomial(poly: list(int), param: int):

		cost	iteration
	count = 0	c_1	1
	max_degree = len(poly) - 1	c_2	1
	max_poly_param = pow(param, max_degree)	c_3	1 (recursive)
	↳ defined below		
	for i in range(0, len(poly)):	c_4	n
	count ± poly[i] * max_poly_param	c_5	n
	max_poly_param /= param	c_6	n
	return count	c_7	1

(AUXILLARY FUNC) base power

```
procedure pow(x: int, n: int):
    if n = 0:
        return 1
    else:
        return (x * pow(x, n-1))
```

for pow, $T(n) = \begin{cases} \Theta(1) & \text{if } n = 0 \\ \underbrace{\Theta(1)}_{\text{splitting cost}} + \underbrace{T(1)}_{\text{left}} + \underbrace{T(n-1)}_{\text{right}} & \text{otherwise} \end{cases}$

(3)

$n=1$: $T(1) = \Theta(1) = c_1$

$n=2$: $T(2) = c_2 + T(1) + T(1) = 2c_1 + c_2$

$n=3$: $T(3) = c_2 + T(1) + T(2) = 3c_1 + 2c_2$

$n=4$: $T(4) = c_2 + T(3) + T(1) = 4c_1 + 3c_2$

$\vdots$

for n: $T(n) = nc_1 + (n-1)c_2 = \Theta(n)$

$\therefore T(n) = O(n)$

A0149963M.

CSC 2229 E2: Assignment 2

Now, proving our axillary function has finished. We need to prove that the run time of compute_polynomial is bounded by $(O(n))$

(3) $T(n) = c_1 + c_2 + c_3 + n*(c_4 + c_5 + c_6) + c_7$

$= (c_1 + c_2 + c_7) + n(c_4 + c_5 + c_6) + \Theta(n)$

$c_3 = \Theta(n)$ because we proved the runtime of the axillary function pow is $\Theta(n)$ previously.

$= (c_1 + c_2 + c_7) + n(c_4 + c_5 + c_6 + c_8)$

Let $c_3 = n c_8$, where $c_8 =$ constant.

$= c_9 + n c_{10}$.

let $c_9 = c_1 + c_2 + c_7$ and $c_{10} = c_4 + c_5 + c_6 + c_8$

(4) Guess $T(n) = O(n)$.

(5). $\therefore$ for some positive constant a,

$c_9 + n c_{10} \leq a n$.

$n(a - c_{10}) \geq c_9$.

$n(a - c_{10}) \geq n c_9 \geq c_9$.

TRUE if $n \geq 1$ and $a - c_{10} \geq c_9$.

$a \geq c_9 + c_{10}$.

$= c_1 + c_2 + c_7 + c_4 + c_5 + c_6 + c_8$

, where c_8 is $\frac{\Theta(n)}{n}$ of the axillary power function.

A0149963M.

YSC 2229 E2 : Assignment 2.

Part 2, Qn 1.

Ranking ; from most efficient to least efficient:

$$1000000, \log n, n, n \log n, n^2, 2^n.$$

Proof that log n less efficient than 1000 000

* Proofs are not done in induction, but by inequality checking and checking values for such inequality holds.

$1000000 = O(\log n)$.

$1000000 \leq a \log_2 n$.

Set $a = 1000000$,

$1000000 \leq 1000000 \log_2 n$.

$\log_2 n \geq 1$.

$n \geq 2$.

$1000000 \leq a \log_2 n$.

The inequality holds for all $n \geq 2$. Therefore, $1000000 = O(\log n)$. holds because it obeys big-O notation standards, where $n_0 = 2$. when $a = 10^6$.

Proof that. ~~log~~ n less efficient than. log n.

$\log n = O(n)$.

* I will use log base 2 ($\log_2$) all the time.

$\log_2 n \leq cn$.

choose $c = 1$, then,

$\log_2 n \leq n$.

$n \leq 2^n$.

This inequality holds for all $n \geq 1$ (since n must be positive). Therefore, $\log n = O(n)$ holds because it obeys big-O, where chosen $c = 1$ and $n_0 = 1$.

Recap: $O(g(n)) = \{ f(n)$: there exists positive constants. c and n_0 such that $0 \leq f(n) \leq c(g(n))$ for all $n \geq n_0 \}$

A0149963M

YSC2229 E2 Assignment 2.

Proof that $n \log n$ less efficient than n.

$$n = O(n \log_2 n).$$
$$n \leq c n \log_2 n.$$
choose $c = 1$. then,
$$n \leq n \log_2 n.$$

Since, n must be positive, dividing by n throughout the inequality will not change the sign. Hence,

$$\log_2 n \geq 1 \Rightarrow n \geq 2.$$

This inequality holds for all $n \geq 2$. Therefore, $n = O(n \log_2 n)$, where chosen $c = 1$ (constant), and $n_0 = 2$ (resultant).

Proof that n^2 less efficient than $n \log n$.

$$n \log n = O(n^2).$$
$$n \log_2 n \leq c n^2.$$
choose $c = 1$, then,
$$n \log_2 n \leq n^2.$$

Because n is positive, dividing both sides by n does not change the sign. Hence,

$$\log_2 n \leq n.$$
$$n \leq 2^n.$$

For positive constants n, this inequality holds for all $n \geq 1$. Therefore, $n \log n = O(n^2)$, where ~~c = 1~~ a chosen $a = 1$ and $n_0 = 1$ (resultant).

A0149963M.

YSC2229 E2 Assignment 1

Date No.

Prove that 2^n less efficient than n^2.

~~2~~. $n^2 = O(2^n)$

$n^2 \leq c\, 2^n$.

choose $c = 1$, then for all $n \geq n_0$, $n_0 \in \mathbb{Z}^+$.

$n^2 \leq 2^n$.

we wish to prove this by induction.

Base case: choose $n = 4$. Then,

$LHS = 16$, $RHS = 2^4 = 16$.

Hence, the statement and inequality holds.

Induction hypothesis:

for some $n \in \mathbb{Z}^+$, $n \geq 4$,

$n^2 \leq 2^n$. (IH).

we wish to show that

$(n+1)^2 \leq 2^{n+1}$:

holds $\forall n \in \mathbb{Z}^+$, $n \geq 4$.

$(n+1)^2 = n^2 + 2n + 1$,

$(2^{n+1}) = 2(2^n) = 2^n + 2^n$.

By induction, choosing ~~when~~ $c = 1$, $n^2 = O(2^n)$ where $n_0 = 4$.

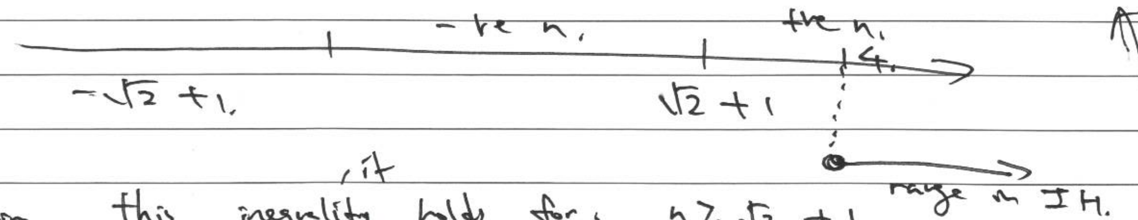
From induction hypothesis, we know that.

$n^2 \leq 2^n$.

Now, consider the inequality. $\boxed{n^2 \geq 2n + 1}$.

$n^2 - 2n - 1 \geq 0$.

completing the square, $(n-1)^2 - 2 \geq 0$.



solving this inequality, it holds for $n \geq \sqrt{2} + 1$

and $n \leq -\sqrt{2} + 1$ (out of range).

$\therefore$ $\forall n \in \mathbb{Z}^+$, $n \geq 4$, $2n + 1 \leq n^2$. But $n^2 \leq 2^n$ (IH).

$\therefore$ Therefore, $2n + 1 \leq 2^n$.

$\therefore$ $(n+1)^2 = n^2 + (2n+1) \leq 2^n + 2^n = 2(2^n)$ is TRUE.

Besform®

A0149963M.

YSC 2229 E2 Assignment 1

2. Disprove $2^{2n} = O(2^n)$.

Assume ~~there~~ that exists constant c, $n \geq n_0$ such that the statement holds. Therefore,

$$2^{2n} \leq c(2^n).$$

$$(2^n)^2 \leq c(2^n).$$

$$c \geq 2^n.$$

However, we arrive at a contradiction here. This is because c is defined to be a constant that does not depend on n. However, for $2^{2n} = O(2^n)$, we need $c \geq 2^n$, which means c is not constant. By contradiction, $(2^{2n}) \neq O(2^n)$.