Algorithm Analysis ~ 05

CSE/IT 122 ~ Algorithms & Data Structures

Review: Single For Loop

How many steps are required to complete the below code snippet?

```
for i = 3 to n - 1
    a = 3 * n + 2 * i - 1
```

Review: Single For Loop

```
for i = 3 to n - 1
    a = 3 * n + 2 * i - 1
```

Answer: for loop fencepost in () and +1 for test:

(n - 1 - 3 + 1) + 1 = n - 2

Inside loop n - 3 times (just the fencepost portion)

So *num_steps* = $c_1(n-2)+c_2(n-3)$

Nested Loops

- ➔ Analyze from the inside out.
- ➔ Total running time is the *product* of the sizes of all loops

Example: Two For Loops

	Cost	Number of Times
for(i = 0; i < n; i++)	c_1	$n + 1$
for (j = 0; j < n; j++)	c_2	$n(n + 1) = n^2 + n$
sum += i;	c_3	n^2

$T(n) = c_1(n + 1) + c_2(n^2 + n) + c_3n^2$

Take A Closer Look

Do this loop explicitly … what happens?

```
count = 0
for i = 1 to 3
    for j = 1 to 3
        print i, j
        count += 1
        print count
```

Take A Closer Look

```
count = 0
for i = 1 to 3
    for j = 1 to 3
        print i, j
        count += 1
        print count
```

```
i = 1      i = 2      i = 3
1, 1       2, 1       3, 1
1          4          7
1, 2       2, 2       3, 2
2          5          8
1, 3       2, 3       3, 3
3          6          9
```

More Examples: You Try

1)

```
for i := 1 to n
    for j := 1 to 2n
        a := 2*n + i * j
```

2)

```
for k := 2 to n
    for j: 1 to 3*n
        x := a[k] - b[j]
```

3)

```
for i:= 1 to n
    s := 0
    for j:= 1 to i
        s:= s + a[j]
    t := t + s^2
```

More Examples: You Try

Code	Cost	# of Times
`for i := 1 to n`	c_1	(n-0-1+1)+1 = n+1
`    for j := 1 to 2n`	c_2	(n)(2n-0-1+1)+1 = $2n^2$ + n
`        a := 2*n + i * j`	c_3	(n)(2n-0-1+1) = $2n^2$

$T(n) = c_1(n+1) + c_2(2n^2+n) + c_3(2n^2)$

More Examples: You Try

Code	Cost	# of Times
`for k := 2 to n`	c_1	$(n-0-2+1)+1 = n$
`   for j := 1 to 3n`	c_2	$(n-1)(3n-0-1+1)+1 = 3n^2-2n-1$
`      x := a[k] - b[j]`	c_3	$(n-1)(3n-0-1+1) = 3n^2-3n$

$$T(n) = c_1(n) + c_2(3n^2-2n-1) + c_3(3n^2-3n)$$

More Examples: You Try

Code	Cost	# of Times
`for i := 1 to n`	c_1	(n - 0 -1 +1)+1 = n+1
`   s := 0`	c_2	n-0-1+1 = n
`   for j := 1 to i`	c_3	1+2+...+(n+1) = n((n+1)/2)
`      s := s +a[j]`	c_4	1+2+...+n = n(((n+1)/2) - 1)
`   t := t + s`^{2}	c_5	n−0+1−1 = n

$T(n) = c_1(n+1) + c_2(n) + c_3((n(n+1)/2) + c_4(n(n+1)/2)-1 + c_5(n)$

Three Four Loops

	Cost	Number of Times
for(i = 0; i < n; i++)	c_1	$n+1$
for (j = 0; j < n; j++)	c_2	$n(n+1) = n^2 + n$
for (k = 0; k < n; k++)	c_3	$n \cdot n \cdot (n+1) = n^3 + n^2$
sum += i;	c_4	n^3

$T(n) = c_1(n+1) + c_2(n^2+n) + c_3(n^3+n^2) + c_4n^3$

Three Loops That Depend on i and J

```
for i = 1 to n
    for j = 1 to i
        for k = 1 to j
            x = i * j * k
```

How to Solve?

Permutations and Combinations

- Definition: a **permutation** of a set of objects is an ordering of the objects in a row.
  - For example: the set of elements a, b, and c have 6 permutations:
    - abc
    - acb
    - cba
    - bac
    - bca
    - cab
  - For example: the set {0,1} has two permutations:
    - 01
    - 10

Permutations

- ➔ Theorem: For any integer $n >= 1$, the number of permutations of a set with n elements is n!
- ➔ Example Six diplomats are to be seated at a circular table. How many ways can you arrange them around a table?

Permutations

- ➔ Theorem: For any integer n >= 1, the number of permutations of a set with n elements is n!
- ➔ Example Six diplomats are to be seated at a circular table. How many ways can you arrange them around a table?
  - Solution: Two seatings are the same if they are just a rotation of each other. Call the diplomats A, B, C, D, E, F and fix A at one of the seats. Then you have to arrange B, C, D, E, F or 5! = 120 ways.

Permutations

➔ Definition: an **r-permutation** of a set of n elements is an ordered selection of r elements taken from the set of n elements. The number of r-permutations of a set of n elements is denoted by P(n,r) and

$$P(n,r) = \frac{n!}{(n-r)!}$$

Permutations: Examples

- Example: How many different three letter words can you form from the english alphabet without repetition?
  - This includes nonsense words.

Permutations: Examples

- Example: How many different three letter words can you form from the english alphabet without repetition?
  - This includes nonsense words.
  - Solution:

$$P(26,3) = \frac{26!}{(26-3)!} = \frac{26 \cdot 25 \cdot 24 \cdot 23!}{23!} = 15600$$

Permutations: Examples

- ➔ Example: How many different 3 letter words begin with the letter z, without repetition?
  - Hint: You will have a fixed z

Permutations: Examples

- Example: How many different 3 letter words begin with the letter z, without repetition?
  - Hint: You will have a fixed z
  - Solution:

$$P(25,2) = \frac{25!}{(25-2)!} = \frac{25 \cdot 24 \cdot 23!}{23!} = 600$$