In-Class Exercise

1. Write a pseudocode algorithm that finds the largest number in a list of numbers.
2. Write a pseudocode algorithm that prints out all the subsets of three elements of a set of n elements. The elements of this set are stored in a list that is a parameter to the algorithm.

Answers

1. Write a pseudocode algorithm that finds the largest number in a list of numbers.

```
number findMax(int n, const keytype S[])
   index i;
   number max = S[1];
   for(i = 2; i <= n; i++)
      if(S[i] > max);
         max = S[i];
   return max;
```

Answers

2. Write a pseudocode algorithm that prints out all the subsets of three elements of a set of *n* elements. The elements of this set are stored in a list that is a parameter to the algorithm.

```
void subsets(int n, const keytype S[])
    index i, j, k;
    for(i = 1; i <= n; i++)
        for(j = i + 1; j <=n; j++)
            for(k = j + 1; k <= n; k++)
                cout << S[i] << S[j] << S[k] << endl;
```

In-Class Exercise

1. How many times faster is gcd(31415, 14142) by Euclid's algorithm than by algorithm 2 provided in the slides? (You'll want to use a calculator!) Compare the two based on the frequency count of the total number of mods performed.
2. What is the time complexity of the Matrix Multiplication algorithm previously given?

```
void MatrixMult(int n, const number A[][], const number B[][],
                number C[][])
{
    index i, j, k;

    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            C[i][j] = 0;
            for (k = 1; k <= n; k++)
                C[i][j] = C[i][j] + A[i][k] * B[k][j];
}
```

In-Class Exercise

3. Consider the following algorithm for finding the distance between the two closest elements in an array of numbers. Make as many improvements as you can:

```
procedure MinDistance(int n, A[])
{
    minDist = ∞;
    for i = 0 to n - 1 do
        for j = 0 to n - 1 do
            if (i != j) and (|A[i] - A[j]| < minDist)
                minDist = |A[i] - A[j]|
    return minDist;
}
```

Answers

1. How many times faster is gcd(31415, 14142) by Euclid's algorithm than by algorithm 2 provided in the slides? (You'll want to use a calculator!) Compare the two based on the frequency count of the total number of mods performed.

The # of divisions made by Euclid's algorithm is 11: gcd(31415, 14142), gcd(14142, 3131),
gcd(3131, 1618), gcd(1618, 1513), gcd(1513, 105), gcd(105, 43), gcd(43, 19), gcd(19, 5), gcd(5, 4), gcd(4, 1), gcd(1, 0)

Since we have now determined that the gcd is 1, we know that the iterative algorithm (algorithm 2 in the slides) takes 14142 iterations! In each iteration, the algorithm makes either 1 or 2 comparisons. Therefore, Euclid's algorithm is between 1 * 14142/1 = **1300** and 2 * 14142/11 **2600** times faster.

Answers

```
void MatrixMult(int n, const number A[][], const number B[][],
               number C[][])
{
    index i, j, k;

    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            C[i][j] = 0;
            for (k = 1; k <= n; k++)
                C[i][j] = C[i][j] + A[i][k] * B[k][j];
}
```

Basic Operation: multiplication instruction in the innermost for loop.

No way to break out of any loop, so we have n times in the *i* loop, *n* times in the *j* loop, *n* times in the *k* loop, or $n * n * n = n^3$

Answers

```
procedure MinDistance(int n, A[])
    minDist = ∞;
    for i = 0 to n - 1 do
        for j = 0 to n - 1 do
            if (i != j) and (|A[i] - A[j]| < minDist)
                minDist = |A[i] - A[j]|
    return minDist;
```

```
procedure MinDistanceBetter(int n, A[])
    minDist = ∞;
    for i = 0 to n - 1 do
        for j = i + 1 to n - 1 do
            int curDist = |A[i] - A[j]|;
            if (curDist  < minDist)
                minDist = curDist
    return minDist;
```

Since we take the absolute value of A[i] - A[j], we don't need to check A[j] - A[i]. Therefore, we can start the j loop from j = i + 1 rather than j = 0, so we avoid double checking.

We create a variable, curDist, to hold the result of |A[*i*] - A[*j*]| so we don't need to calculate it twice.

Since we are now starting the j loop at j = i + 1, we no longer need the (*i* != *j*) check in the if statement.

In-Class Exercise

1. $f(x) = 3n^2 + 10n\lg n + 1000n + 4\lg n + 9999$ is in which complexity class?

   $\Theta(\lg n)$, $\Theta(n^2\lg n)$, $\Theta(n)$, $\Theta(n\lg n)$, $\Theta(n^2)$

2. Determine whether the following are in the same complexity class:
   a. $2^{\lg n}$ and n
   b. $n^{1/2}$ and $(\lg n)^2$
   c. $(n - 1)!$ and $(n)!$
3. Prove that $6n^2 + 20n \in O(n^2)$

Answers

1. $f(x) = 3n^2 + 10n\lg n + 1000n + 4\lg n + 9999$ is in which complexity class?

   $\Theta(n^2)$ is n^2 is the dominant term.

In-Class Exercise

2. Determine whether the following are in the same complexity class:

$2^{\lg n}$ and n

$2^{\lg n} = n^{\lg 2} = n$ (since $\lg 2 = 1$)

Therefore, they are in the same complexity class.

Answers

2. Determine whether the following are in the same complexity class:

$n^{1/2}$ and $(\lg n)^2$

Always take the lg of both sides to remove exponents!

We are left with:

$\lg n^{1/2}$ and $\lg(\lg n)^2$

$\frac{1}{2}(\lg n)$ and $2\lg\lg n$

Remove constants, and $\lg n$ grows faster than $\lg\lg n$

Answers

2. Determine whether the following are in the same complexity class

(n - 1)! and (n)!

No. (n-1)! grows slower than (n)! Remember, if we have 100n and n and n = 1, 100 is 100 times larger than 1. If we have n = 3, 300 is still 100 times larger than 3.

However, with (n - 1)!, each time n grows, (n - 1)! is n less than n! Therefore, they don't grow at the same rate.

In-Class Exercise

3. Prove that $6n^2 + 20n \in O(n^2)$

$6n^2 + 20n >= cn^2$ when $n >= N$

When does $6n^2$ start to dominate $20n$?

When $n = 4$.

Plug 4 in:

$96 + 80 >= c16$

$c = 11$ and $N = 4$