CSC 215

Math and Computer Science

Permutations

- Heart of many brute force algorithms
- Generate all possible combinations
- How many are there?

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

$$n = 3, k = 3$$

$$P(3,3) = \frac{n!}{(n-k)!} = 3! = 6$$

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

P(3,3)=3!=6

- Set {0,1,2} , n = 3

  {0,1,2} {0,2,1} {1,0,2} {1,2,0} {2,0,1} {2,1,0}

  6 ways to order the 3 numbers

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

P(4,4) = 4! = 24

- Set {0,1,2,3}

  {0,1,2,3} {0,1,3,2} {0,2,1,3} {0,2,3,1} {0,3,1,2} {0,3,2,1}
  {1,0,2,3} {1,0,3,2} {1,2,0,3} {1,2,3,0} {1,3,0,2} {1,3,2,0}
  {2,0,1,3} {2,0,3,1} {2,1,0,3} {2,1,3,0} {2,3,0,1} {2,3,1,0}
  {3,0,1,2} {3,0,2,1} {3,1,0,2} {3,1,2,0} {3,2,0,1} {3,2,1,0}

Solving P(3,3) Iteratively

```
void iter_permute3()
{
    int i, j, k;
    for ( i = 0; i < 3; i++ )
        for ( j = 0; j < 3; j++ )
            for ( k = 0; k < 3; k++ )
                if ( i != j && i != k && j != k )
                {
                    cout << i << " " << j << " " << k << endl;
                }
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Solving P(4,4) Iteratively

```
void iter_permute4()
{
   int i,j,k,l;

   for(i=0; i<4; i++)
      for(j=0; j<4; j++)
         for(k=0; k<4; k++)
            for(l=0; l<4; l++)
               if( i!=j && i!=k && i!=l && j!=k && j!=l && k!=l )
                  cout << i << " " << j << " " << k << " " << l << endl;

}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Solving P(10,10) Iteratively

```
void iter_permute10()
{
   int i,j,k,l,m,n,o,p,q,r;
   for(i=0; i<10; i++)
      for(j=0; j<10; j++)
         for(k=0; k<10; k++)
            for(l=0; l<10; l++)
               for(m=0; m<10; m++)
                  for(n=0; n<10; n++)
                     for(o=0; o<10; o++)
                        for(p=0; p<10; p++)
                           for(q=0; q<10; q++)
                              for(r=0; r<10; r++)
if( i!=j && i!=k && i!=l && i!=m && i!=n && i!=o && i!=p && i!=q && i!=r &&
    j!=k && j!=l && j!=m && j!=n && j!=o && j!=p && j!=q && j!=r &&
    k!=l && k!=m && k!=n && k!=o && k!=p && k!=q && k!=r &&
    l!=m && l!=n && l!=o && l!=p && l!=q && l!=r &&
    m!=n && m!=o && m!=p && m!=q && m!=r &&
    n!=o && n!=p && n!=q && n!=r &&
    o!=p && o!=q && o!=r &&
    p!=q && p!=r &&
    q!=r )
          cout << i << " " << j << " " << k << " " << l << " " << m << " "
               << n << " " << o << " " << p << " " << q << " " << r << endl;
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Iterative Approach

- n must be known before programming
- n number of for loops will be needed
- If statement to make sure no two numbers are the same.

Recursive Approach

- N not necessarily known at time of programming
- Each recursive call will nest a for loop inside of another for loop
- We will not put duplicate numbers into our solution

What is Needed – Set of n Integers

- An array of size (n) to hold the solution
- An array to keep track of the numbers used in the solution
- A variable to keep track of how many numbers are in the solutions so far
- A variable to store the size of our set (n)

The Solution Array

- Assume n = 3
- int p[3];

P Array	2	0	1
Index	0	1	2

- The P array holds a solution set of {2,0,1}

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

The Used Array

- Lets look a placing a number someplace in the P array
- Put the number 2 in the zeroth spot of the P array

P	2		
Index	0	1	2

Used	0	0	1
Index	0	1	2

- Mark the 2 spot of the used array with a 1 to show it is in the solution already

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Other Variables

- Need a variable to keep track of how many numbers we have put into the P array.
  - Tells us we have inserted k number of n
  - Tells us the location (index) of where to put the next number in the P array.
  - Call it pos for position
- Need a variable to tell us how many items are in the set.
  - Call it n,

Filling a Spot in P

- n = 10
- pos = 4 <= some spot in the P array

P	x	x	x	x	?					
Index	0	1	2	3	4	5	6	7	8	9

- Must try and put every number in this location

```
for( i=0; i<n; i++)
```

- There are numbers in the previous spots and we can not duplicate.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Filling a Spot in P - continued

P	x	x	x	x	?					
Index	0	1	2	3	4	5	6	7	8	9

```
for( item=0; item<n; item++)
{
    if( used[item] == 0 )
    {
        p[pos] = item;              // move item into solution
        used[item] = 1;             // mark the item as used
        // fill the other spots
        used[item] = 0;             // mark the item as unused and continue
    }
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Moving to the next position

P	x	x	x	x	?					
Index	0	1	2	3	4	5	6	7	8	9

```
for( item = 0; item < n; item++)
{
    if( used[item] == 0 )
    {
        p[pos] = item;                  // move item into solution
        used[item] = 1;                 // mark the item as used
        permute( p, used, n, pos+1 ); // fill the other spots
        used[item] = 0;                 // mark the item as unused and continue
    }
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

The Base Case

- When I have put n number of items in the P array, we have a solution.
- When this happens, do something with the solution, like print it out.

```
if( pos == n )
{
    for( i=0; i<n; i++)
        cout << p[i] << " ";
    cout << endl;
    return;    // back up to the previous function
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 0

P			
Index	0	1	2

Used	0	0	0
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 0          ← move into P at pos, mark used
                                    make recursive call to pos+1
```

P	0		
Index	0	1	2

Used	1	0	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	0		
Index	0	1	2

Used	1	0	0
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 1          ← already in the solution
Item = 1, used[item] = 0          ← move into P at pos, mark used
                                     make recursive call to pos+1
```

P	0	1	
Index	0	1	2

Used	1	1	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	0	1	
Index	0	1	2

Used	1	1	0
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 1          ← 0 is in solution already
Item = 1, used[item] = 1          ← 1 is in solution already
Item = 2, used[item] = 0          ← move into P at pos, mark used
                                     make recursive call to pos+1
```

P	0	1	2
Index	0	1	2

Used	1	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 3

P	0	1	2
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
Base case is reached (pos == n)
print out solution to screen.

                                        0 1 2

Return to try other numbers
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	0	1	2
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← 0 is in solution already
Item = 1, used[item] = 1        ← 1 is in solution already
Item = 2, used[item] = 0        ← Mark 2 as unused
Item = 3,                       ← return, tried all numbers
```

P	0	1	
Index	0	1	2

Used	1	1	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	0	1	
Index	0	1	2

Used	1	1	0
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← 0 is in solution already
Item = 1, used[item] = 0        ← Mark 1 as unused
Item = 2, used[item] = 0        ← move into P at pos, mark used
                                   make recursive call to pos+1
```

P	0	2	
Index	0	1	2

Used	1	0	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	0	2	
Index	0	1	2

Used	1	0	1
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 1          ← 0 is in solution already
Item = 1, used[item] = 0          ← move into P at pos, mark used
                                     make recursive call to pos+1
```

P	0	2	1
Index	0	1	2

Used	1	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 3

P	0	2	1
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
Base case is reached (pos == n)
print out solution to screen.
                                                    0 1 2
Return to try other numbers                         0 2 1
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	0	2	1
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← 0 is in solution already
Item = 1, used[item] = 0        ← Mark 1 as unused
Item = 2, used[item] = 1
Item = 3                        ← return
```

P	0	2	
Index	0	1	2

Used	1	0	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	0	2	
Index	0	1	2

Used	1	0	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← 0 is in solution already
Item = 1, used[item] = 0        ← Mark 1 as unused
Item = 2, used[item] = 0        ← Mark 2 as unused
Item = 3                        ← return
```

P	0		
Index	0	1	2

Used	1	0	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 0

P	0		
Index	0	1	2

Used	1	0	0
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 1        ← move into P at pos, mark used
                                  make recursive call to pos+1
```

P	1		
Index	0	1	2

Used	0	1	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	1		
Index	0	1	2

Used	0	1	0
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 1          ← move into P at pos, mark used
                                    make recursive call to pos+1
```

P	1	0	
Index	0	1	2

Used	1	1	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	1	0	
Index	0	1	2

Used	1	1	0
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 1          ← 0 is in solution already
Item = 1, used[item] = 1          ← 1 is in solution already
Item = 2, used[item] = 0          ← move into P at pos, mark used
                                    make recursive call to pos+1
```

P	1	0	2
Index	0	1	2

Used	1	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 3

P	1	0	2
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
Base case is reached (pos == n)
print out solution to screen.
```

```
0 1 2
0 2 1
1 0 2
```

```
Return to try other numbers
```

Walk through with P(3,3) – pos = 2

P	1	0	2
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← 0 is in solution already
Item = 1, used[item] = 1        ← 1 is in solution already
Item = 2, used[item] = 0        ← Mark 2 as unused
Item = 3                        ← return
```

P	1	0	
Index	0	1	2

Used	1	1	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	1	0	
Index	0	1	2

Used	1	1	0
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 1        ← 1 is in solution already
Item = 2, used[item] = 0        ← move into P at pos, mark used
                                   make recursive call to pos+1
```

P	1	2	
Index	0	1	2

Used	0	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	1	2	
Index	0	1	2

Used	0	1	1
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 0          ← move into P at pos, mark used
                                    make recursive call to pos+1
```

P	1	2	0
Index	0	1	2

Used	1	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 3

P	1	2	0
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
Base case is reached (pos == n)
print out solution to screen.
```

```
0 1 2
0 2 1
1 0 2
1 2 0
```

```
Return to try other numbers
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	1	2	0
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 1
Item = 2, used[item] = 1
Item = 3                        ← Return
```

P	1	2	
Index	0	1	2

Used	0	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	1	2	
Index	0	1	2

Used	0	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 1        ← 1 is in solution already
Item = 2, used[item] = 0        ← Mark 2 as unused
Item = 3                        ← return
```

P	1		
Index	0	1	2

Used	0	1	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 0

P	1		
Index	0	1	2

Used	0	1	0
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 0        ← Mark 1 as unused
Item = 2, used[item] = 0        ← move into P at pos, mark used
                                  make recursive call to pos+1
```

P	2		
Index	0	1	2

Used	0	0	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	2		
Index	0	1	2

Used	0	0	1
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 0          ← move into P at pos, mark used
                                    make recursive call to pos+1
```

P	2	0	
Index	0	1	2

Used	1	0	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	2	0	
Index	0	1	2

Used	1	0	1
Index	0	1	2

```
for( item=0; item<n; item++)      // start new for loop
Item = 0, used[item] = 1          ← 0 already in solution
Item = 1, used[item] = 0          ← move into P at pos, mark used
                                     make recursive call to pos+1
```

P	2	0	1
Index	0	1	2

Used	1	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 3

P	2	0	1
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
Base case is reached (pos == n)
print out solution to screen.
```

```
0 1 2
0 2 1
1 0 2
1 2 0
2 0 1
```

```
Return to try other numbers
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	2	0	1
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← 0 already in solution
Item = 1, used[item] = 0        ← Mark 1 as unused
Item = 2, used[item] = 1        ← 2 already in solution
Item = 3                        ← return
```

P	2	0	
Index	0	1	2

Used	1	0	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	2	0	
Index	0	1	2

Used	1	0	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 1        ← Mark 0 as unused
Item = 1, used[item] = 0        ← move into P at pos, mark used
                                  make recursive call to pos+1
```

P	2	1	
Index	0	1	2

Used	0	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	2	1	
Index	0	1	2

Used	0	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0
```

```
// start new for loop
← move into P at pos, mark used
  make recursive call to pos+1
```

P	2	1	0
Index	0	1	2

Used	1	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 3

P	2	1	0
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
Base case is reached (pos == n)
print out solution to screen.
```

```
0 1 2
0 2 1
1 0 2
1 2 0
2 0 1
2 1 0
```

```
Return to try other numbers
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 2

P	2	1	0
Index	0	1	2

Used	1	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 1        ← 1 already in solution
Item = 2, used[item] = 1        ← 2 already in solution
Item = 3                        ← return
```

P	2	1	
Index	0	1	2

Used	0	1	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 1

P	2	1	
Index	0	1	2

Used	0	1	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 0        ← Mark 1 as Unused
Item = 2, used[item] = 1        ← 2 already in solution
Item = 3                        ← Return
```

P	2		
Index	0	1	2

Used	0	0	1
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Walk through with P(3,3) – pos = 0

P	2		
Index	0	1	2

Used	0	0	1
Index	0	1	2

```
for( item=0; item<n; item++)
Item = 0, used[item] = 0        ← Mark 0 as unused
Item = 1, used[item] = 0        ← Mark 1 as unused
Item = 2, used[item] = 0        ← Mark 2 as unused
Item = 3                        ← return
```

P			
Index	0	1	2

Used	0	0	0
Index	0	1	2

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY