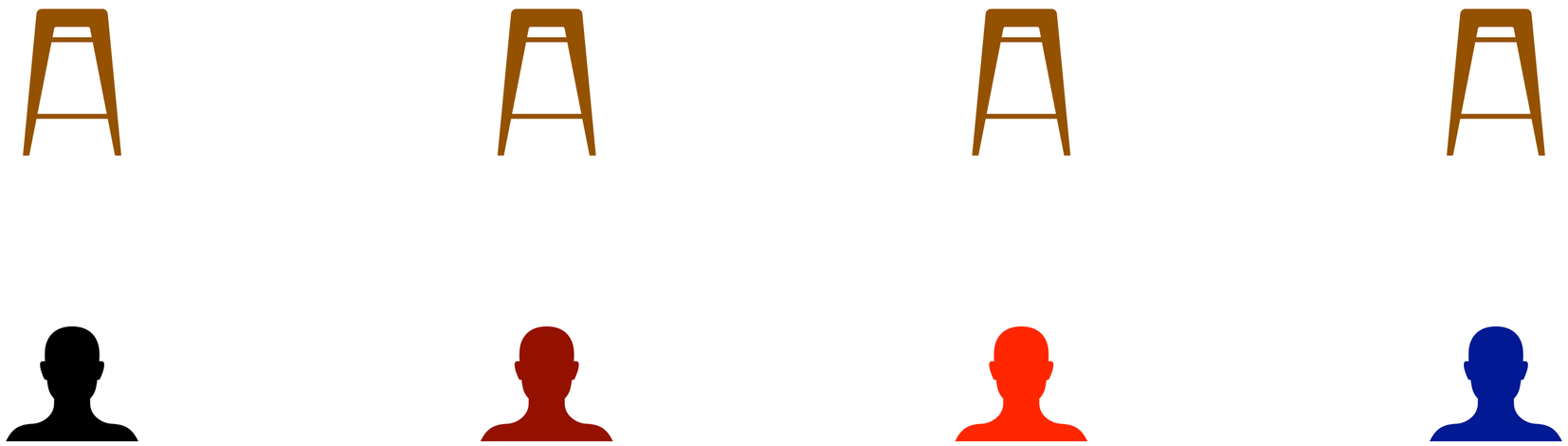
NUMBER THEORY: COUNTING

ISIS 2804

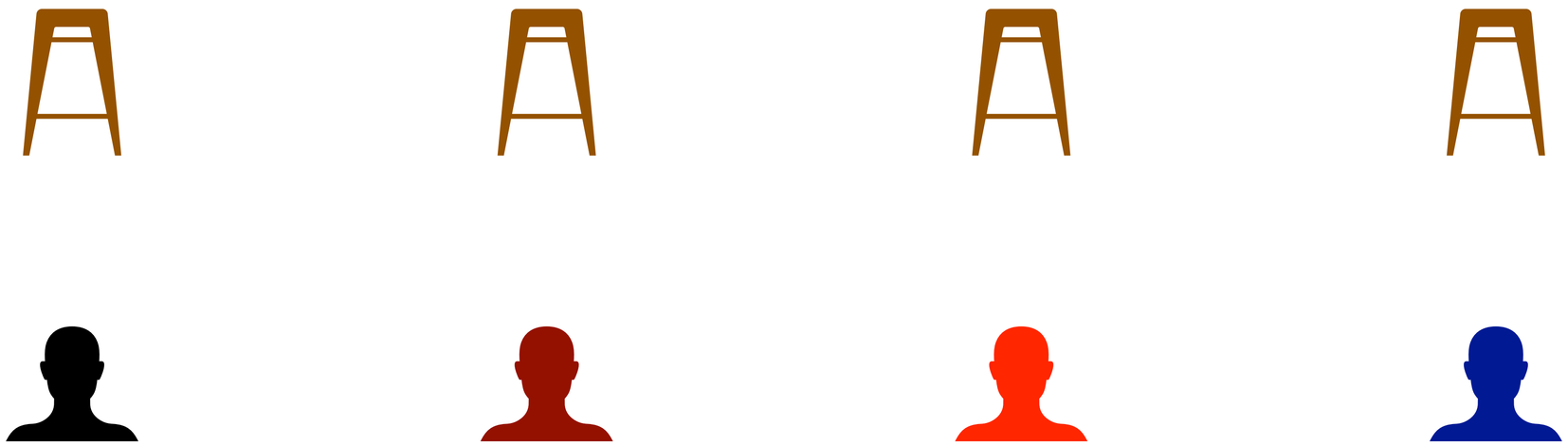
BASICS

Ways to organize n people



Ways to organize n people

Ways to organize n people



Ways to organize n people

Ways to organize n people

For each chair, I can put any of the people. However, once one person sits down, you have one less person to sit, and so on …

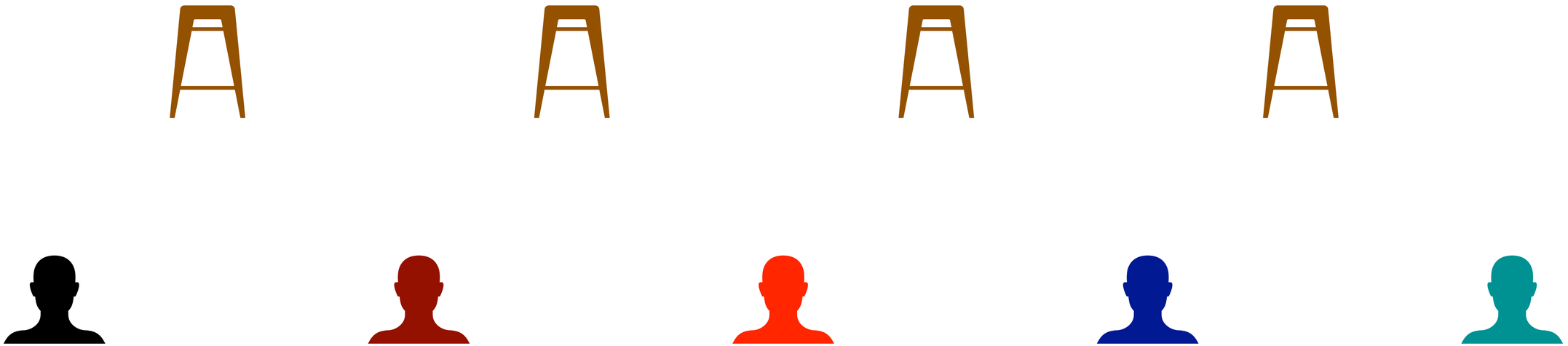
n n–1 n–2 n–3

ways to organize n people: n!

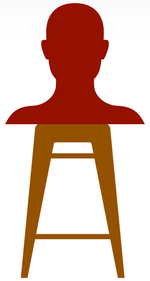
Ways to organize n people

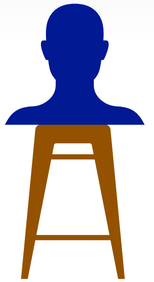
```
fun fact(n : Int) : Int {
    tailrec fun factRec(num : Int, accum : Int) : Int {
      if(num <= 1)
        return accum
      else
        return factRec(num-1, num*n)
    }
    return factRec(n, 1);
}
```

Ways to select k out of n people

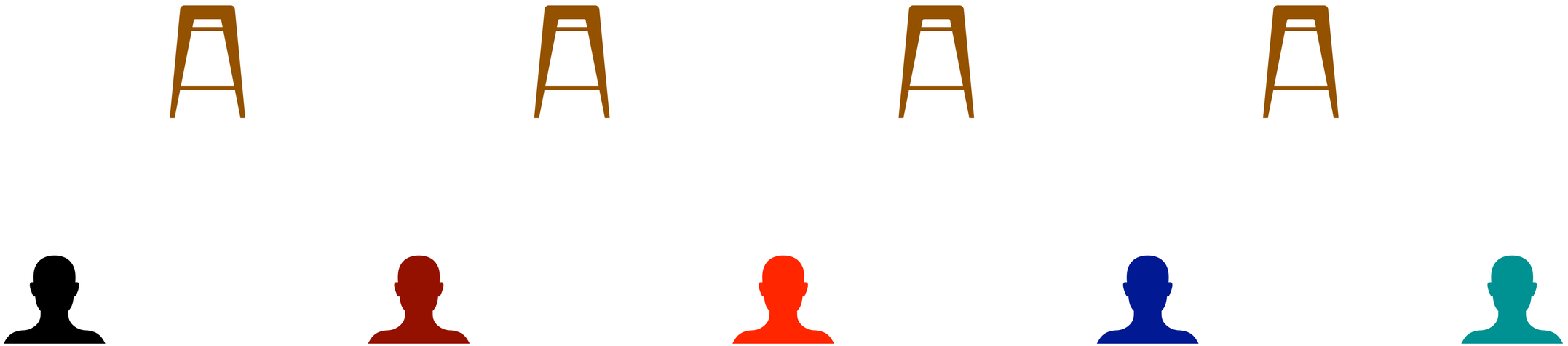


Ways to select k out of n people

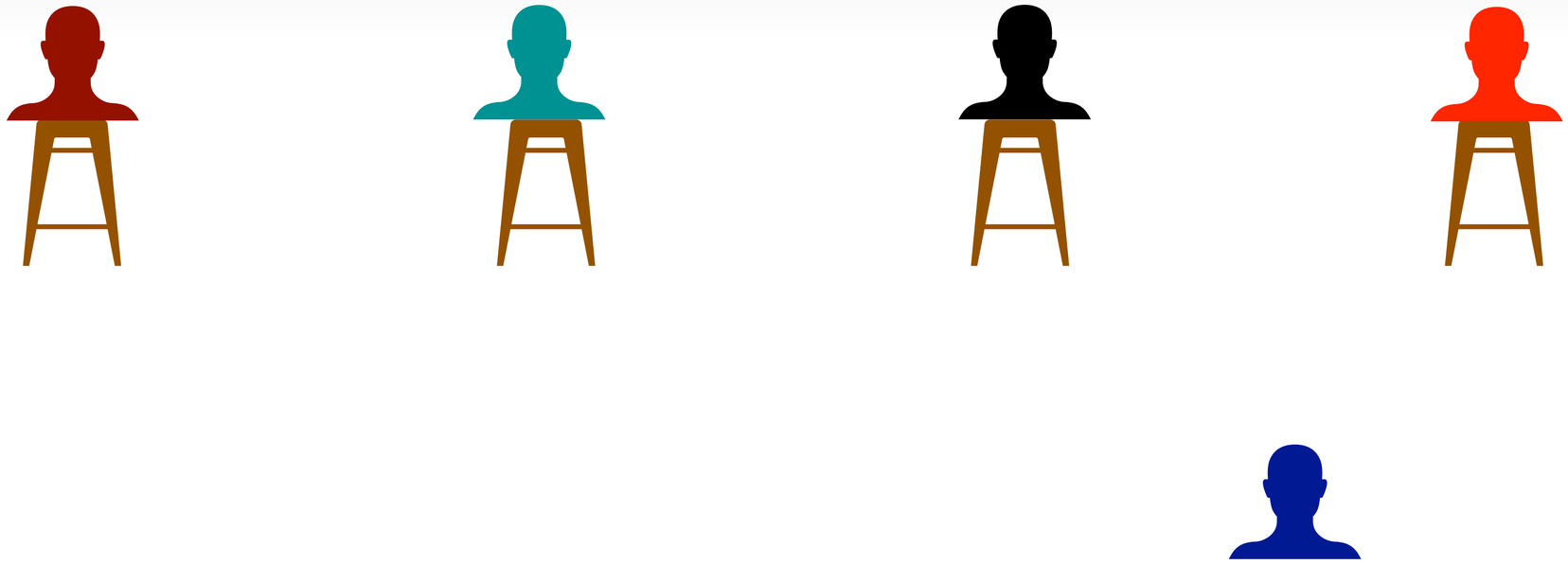






Ways to select k out of n people



Ways to select k out of n people



Ways to select k out of n people

For each chair, I can put any of the k people. However, once one person sits down, you have one less person to sit, and so on ...

n n−1 n−2 n−3

ways to **choose** k people from n people:

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

Ways to select k out of n people

```
void combination(int n, int k) {
    cout << fact(n) / (fact(k) * fact(n-k));
}
```

```
vector<int> f(n, 1);
for(int i=2; i<n; i++) {
  f[i] = i * f[i];
}
void combination(int n, int k) {
    cout << f[n] / (f[k] * f[n-k]);
}
```

Derangements

Derangements are permutations in which no element can appear in its original position

- We can place person 1 in chairs `2, 3`, or `4`
- If we put person `1` at chair i, then:
  - person `i` can be placed in chair `1` (and now we solve the same problem but for size `n-2`)
  - person `i` can sit anywhere else (and now we solve the same problem but for size `n-1`)

Derangements

```
vector<int> derangements(n, 0);
derangements[2] = 1;
for(int i=2; i<n; i++) {
  derangements[i] = (i-1)*(derangements[i-1] + derangements[i-2]);
}
int countDer(int n) {
   return derangements[n];
}
```

Derangements

```
int countDer(int n) {
  if (n == 1) return 0;
  if (n == 2) return 1;
  return (n - 1) * (countDer(n - 1) + countDer(n - 2));
}
```

```
vector<int> derangements(n, 0);
derangements[2] = 1;
for(int i=2; i<n; i++) {
  derangements[i] = (i-1)*(derangements[i-1] + derangements[i-2]);
}
int countDer(int n) {
  return derangements[n];
}
```