演算法與程式解題實務

Mong-Jen Kao (高孟駿)

Monday 18:30 – 21:20

使用遞迴來做搜尋

前言

- 遞迴 (Recursion) 是一個強大的程式撰寫技巧,
  時常可以讓我們用很精簡又有效率的方式解決問題。
- 這個主題的目標, 是要使用**遞迴**來產生「排列/組合的所有可能」
  - 組合 ex: { 1, 2, ..., n } 的所有子集合
  - 排列 ex: 1 到 n 的所有排列
- 想法：
  - 有 n 個決定要做, 「每次遞迴裡, 做一個決定, 然後往下遞迴, 產生此情況下所有的可能」

產生 { 1,2,...,n } 的所有子集合

產生 { 1,2,...,n } 的所有子集合

- 共有 n 個選擇要做決定 (每個數字 **選**或**不選**)
  - **由左到右, 一次做一個選擇**
  - 做完 n 個選擇, 就得到一個子集合

產生 { 1,2,...,n } 的所有子集合

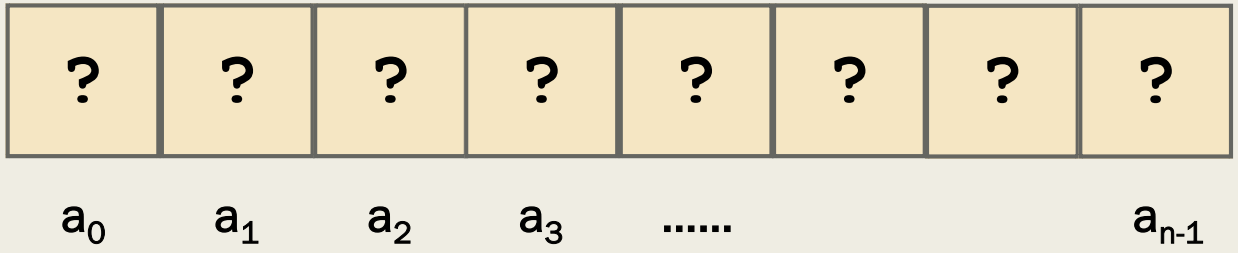

```
void generate_subsets( int k );
```

- 決定 k 要不要選, 再分別遞迴到下一層！
- 遞迴完 n 層, 代表已經做完決定, 得到一個子集合了

```
int mark[30]; // 紀錄數字是否有被選進子集合裡, 1 代表有選

void  generate_subsets( int k, int n )
{
      if( k == n )
            print the subset and return.

      mark[i] = 0;
      generate_subsets(k+1, n);    // 不選 k 的情況, 往下遞迴

      mark[i] = 1;
      generate_subsets(k+1, n);    // 選 k 的情況, 往下遞迴
}
```

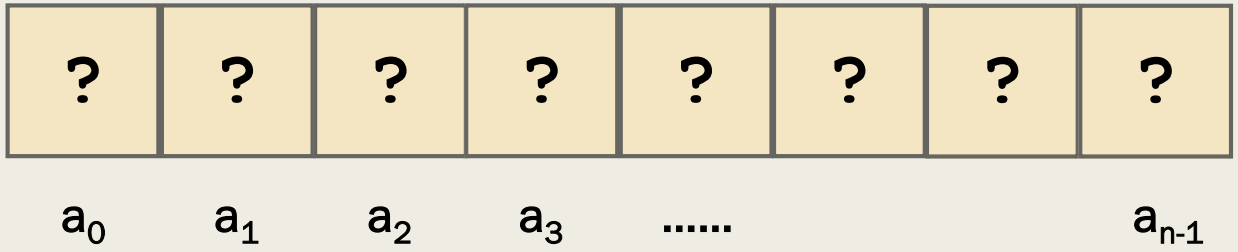
產生 1 到 n 的所有排列

產生 1 到 n 的所有排列



- 共有 n 個位置需要填入數字
  - **由左到右, 一次填入一個數字**
  - 假設前 k 個位置 a_0 a_1, ..., a_{k-1} 已經填好了, 那麼可以填入 a_k 的, 只有尚未使用過的數字
  - 每個選擇都對應到不同的排列

產生 1 到 n 的所有排列




```
void  generate_permut( int k );
```

- 在 a_0 a_1, ..., a_{k-1} 已經填好的情況下,
  考慮所有合法的可能, 分別填入 a_k 然後遞迴！
- 遞迴到第 n 層時, 代表已經填出一個完整的排列.

```
int n, permut[30];
int mark[30]; // 紀錄數字是否已用過, 0 代表未使用

void  generate_permut( int k )
{
      if( k == n )
            print the permutation and return.

      For each i=0,1,…,n-1 with mark[i] == 0   // 考慮每個可以填入的數字
      {
            permut[k] = i;
            mark[i] = 1;

            generate_permut(k+1);    // 遞迴

            mark[i] = 0;
      }
}
```

本週程式題導覽

Problem – A, B

- 給定一個整數方程式,
  計算非負整數解的個數 (產生所有非負整數解)
  - 使用遞迴，一次決定一個變數的值

Problem – C

- 八皇后 (8-Queen) 問題的變型
  - 目標: 將 8 個皇后同時放到棋盤上,
    使她們彼此相安無事, 無法攻擊到對方.
  - 產生所有可能的排法.
- 方法: 使用遞迴, column by column,
  一次決定一個 column 上皇后的位置.