プログラム設計法

胡　振江

分割統治法

- 効率のよいアルゴリズムの設計の有用な方法の一つ

問題Pを幾つかの部分問題（それぞれはPと同類の問題であるが、入力の大きさが小さいもの）に分割し、部分問題の解を集めてもとの問題の解とする手法

整列：挿入整列法

- リストを第一要素と残りの部分に分割して処理を行う。

```
isort = foldr insert []
insert x xs = takeWhile (<=x) xs ++ [x] ++
              dropWhile (<=x) xs
```

T_insert(n) = O(n)
T_isort(n) = T_insert(0) + T_insert(1) + … + T_insert(n)
= O(n^2)

併合整列法

- リストをほぼ同じ大きさの２つの部分に分割し、それぞれの部分を整列したあとで併合する。

```
msort xs | n <= 1    = xs
         | otherwise = merge (msort us) (msort vs)
      where n = length xs
            us = take (n `div` 2) xs
            vs = drop (n `div` 2) xs
merge [] ys = ys
merge (x:xs) [] = x:xs
merge (x:xs) (y:ys) | x<=y      = x : merge xs (y:ys)
                    | otherwise = y : merge (x:xs) ys
```

T_sort(n)
= O(1), if n<=1
= 2T_sort(n/2)+T_merge(n), otherwise

T_merge(n) = O(n)

クイック整列法

- 複雑な分割＋簡単な統合

```
qsort [] = []
qsort (x:xs) = qsort [u|u<-xs,u<x] ++
               [x] ++
               qsort [u|u<-xs,u>=x]
```

T_qsort(n) = O(n^2)　(平均的に O(n log n))

二分探索法

- 問題
  - 整数aと整数 b と述語 p が与えられたとき、区間[a..b]内でp xが成立するような最小の x を求める。
- 仕様

```
find p a b = min [ x | x <- [a..b], p x]
```

  - 正しい：問題の翻訳
  - 効率が悪い　←　b-a+1回の p の計算が必要

効率化１

- [a..b]の単調性質の利用

```
find p a b = min [ x | x <- [a..b], p x]
  (pはb-a+1回評価される)
➔
find p a b = head [ x | x <- [a..b], p x]
        (pは1からb-a+1回評価される)
```

効率化２

- 述語pが単調の場合

p x = True ∧ x < y ならば　p y = True


```
find p a b   | a==b && p a          = a
             | a<b && p m           = find  p a m
             | a<b && not (p m)     = find p (m+1) b
  where
        m = (a+b) `div` 2
```

T(n) = O(log n)

探索と数え上げ

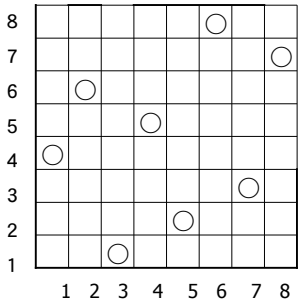
- 組合せ論的問題：ある性質を満たす対象の組合せを探す。
- 手法：逆戻り探索法
- 例題：
  - Eight-queen 問題
  - Instant insanity 問題

エイトクイーン問題

- チェス盤と8個のクイーンが与えられたとき、どの２つのクイーンも互いに効き筋にはならないように盤面に置く。

盤面：
[4,6,1,5,2,8,3,7]

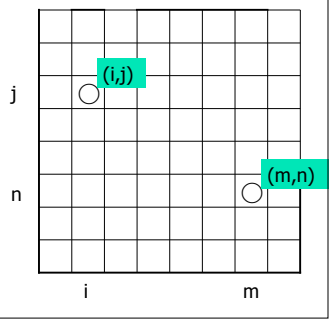
同じの列に二つ以上のクイーンがない。



安全性チェック

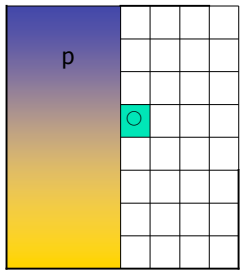
- Check: 座標(i,j)と(m,n)にある２つのクイーンはそれらが同一の行、あるいは２つの対角線上のいずれかにあるか

```
check (i,j) (m,n)
  = j == n  ||
    i+j == m+n ||
    i-j == m-j
```



盤面にクイーンの追加

```
safe p n
  = and [ not (check (i,j) (m,n))
        | (i,j) <- zip [1..length p] p ]
    where m = length p + 1
```



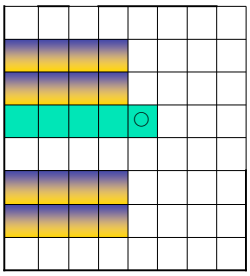
p' = p ++ [n]

メーンプログラム

後戻しのため、最初のm個のクイーンの
さまざまな方法を考える必要がある。

```
queens 0 = [[]]
queens (m+1)
  = [ p ++ [n]
      | p <- queens m,
        n <- [1..8],
        safe p n]

queens1 m = head (queens m)
```



p’ = p ++ [n]

Crypt-arithmetic puzzle

10個までの文字を用いた3個の単語を次のように配置するような問題を解くプログラムを書け。

```
  F O U R
+ F I V E
----------
  N I N E
```

(解の例：2970+2483=5453)

```
puzzle = [ (e,f,i,n,o,r,u,v)
         | e<-[0..9],
           f<-[1..9],
           …,
           v<-[0..9],
           vplus [f,o,u,r] [f,i,v,e] == [n,i,n,e] ]
```

期末試験

- 日時：2月９日
  8：30～10：00まで
- 場所：６３号室
- 教科書、ノートを持ち込み可