2 Divide et impera e Ricorsione

Esercizio 1 Dato un array di interi $A[1..n]$, chiamiamo *gap* un indice $i \in [1, n)$ tale che $A[i+1] - A[i] > 1$.

i. Mostrare per induzione su n che un array $A[1..n]$ tale che $A[n] - A[1] \geq n$ (quindi $n \geq 2$) contiene almeno un gap.

ii. Fornire lo pseudocodice di una procedura ricorsiva divide et impera *gap* che dato un array $A[1..n]$ tale che $A[n] - A[1] \geq n$ restituisce un gap in A.

iii. Valutare la complessità della funzione, utilizzando il master theorem.

Soluzione: La prova per induzione è la seguente:

- $(n = 2)$ Banale, il gap è in $i = 1$.
- $(n > 2)$ Sia $A[n] - A[1] \geq n$. Allora se $A[n] - A[n-1] > 1$, si ha che $i = n - 1$ è un gap e abbiamo concluso. Altrimenti, si osserva che $A[n] - A[1] = A[n] - A[n-1] + A[n-1] - A[1]$. Quindi $A[n-1] - A[1] = (A[n] - A[1]) - (A[n] - A[n-1]) \geq n - 1$, quindi per ipotesi induttiva $A[1..n-1]$ contiene un gap.

La prova funziona anche se invece che spezzare l'array in $A[1..n-1]$ e $A[n-1..n]$, lo si spezza in $A[1..\lceil n/2 \rceil]$ e $A[\lceil n/2 \rceil..n]$. Da questo segue l'algoritmo:

```
gap(A,p,r)    // p < r (almeno due elementi) , A[r] - A[p] >= r-p+1
  if p = r-1
     return p
  else
     q = (p+r)/2
     if A[q] - A[p] > q-p+1
        gap(A,p,q)
     else
        // note che allora A[r] - A[q] > r-q+1
        gap(A,q,r)
```

Si ha che $T(n) = T(n/2) + 1$, quindi il master theorem, dato che $n^{\log_b a} = n^{\log_2 1} = n^0 = \Theta(1)$, ci dice che $T(n) = \Theta(n^0 \log n) = \Theta(\log n)$.

Esercizio 2 Scrivere una procedura di tipo divide et impera *over* che dato un array di interi distinti $A[1..n]$, ordinato in modo crescente, e un intero x restituisce l'indice del più piccolo elemento in A strettamente maggiore di x. Se nessun elemento di A soddisfa la condizione, si restituisca $n + 1$. Valutare la complessità dell'algoritmo.

Soluzione: Si realizza una funzione ricorsiva `over(A,p,r,x)` che restituisce il minimo indice $i \in [p, r]$ tale che $A[i] > x$ se un tale indice esiste, altrimenti $r + 1$.

```
over(A,p,r,x)
    if r-p < 0
       return r+1
    else
       q = floor((r+p)/2)
       if A[q] <= x
          return over(A,q+1,r,x)
       else
          return over(A,p,q-1,x)
```

La correttezza può essere dimostrata per induzione sulla lunghezza n del sottoarray, ovvero $n = |r - p + 1|$. Quando $n = 0$ quindi $r < p$ l'intervallo $[p, r]$ è vuoto, e correttamente la funzione ritorna $r + 1$. Se invece $n > 0$, ovvero $r > p$, sia $q = \lfloor p + r/2 \rfloor$. Distinguiamo due casi:

- se `A[q]>x` allora, per ipotesi induttiva, la chiamata `over(A,p,q-1,x)` ritorna il minimo indice `i` in `[p,q-1]` tale che `A[i]>x`, se un tale indice esiste, e in questo caso è anche il minimo indice in `[p,r]` con questa proprietà. Se nessun elemento in `A[p,q-1]` è maggiore di `x` correttamente ritorna $q - 1 + 1 = q$ che `[p,q-1]` sarà il minimo indice in `[p,r]` di un elemento maggiore di x.
- se `A[q]<=x` allora la chiamata `over(A,q+1,r,x)` ritorna il minimo indice `i` in `[q+1,r]` tale che `A[i]>x`, se un tale indice esiste (e in questo caso sarà anche il minimo indice in `[p,r]`), e $r + 1$ altrimenti.

La complessità è data dalla ricorrenza $T(n) = T(n/2) + c$ e quindi $T(n) = \Theta(\log n)$ dal master theorem.

Esercizio 3 Realizzare una procedure di tipo divide et impera `Max(A,p,r)` per trovare il massimo nell'array `A[p,r]`. Si assuma che l'array non sia vuoto ($\mathtt{p} \leq \mathtt{r}$). Scrivere lo pseudocodice e valutare la complessità con il master theorem.

Soluzione:

```
Max(A, p, r)
    if p = r
       return A[p]
    else // p < r
       q =  (p+r)/2
       m1 = Max(A,p,q)
       m2 = Max(A,q+1,r)
       if m1<m2
                 return m2
       else
          return m1
```

Esercizio 4 Sia `A[1..n]` un array di interi distinti ordinato in senso crescente. Dimostrare che dato un qualunque indice `i`, se `A[i] > i` allora `A[j]>j` per ogni `j>i` e analogamente se `A[i] < i` allora `A[j]<j` per ogni `j<i`.

Utilizzare l'osservazione per realizzare una funzione `Fix(A)` che dato l'array di interi `A[1..n]` ordinato senza ripetizioni restituisce un indice `i` tale che `A[i]=i`, se esiste, e `0` altrimenti. Valutarne la complessità.

Soluzione: Per la prima parte sia i un indice tale che $A[i] > i$, ovvero $A[i] \geq i + 1$. Si osserva che, se $i < n$ allora $A[i+1] > A[i] \geq i+1$ ovvero $A[i+1] > i+1$. Usando questo fatto, si conclude con un ragionamento induttivo (che per ogni $j \geq 1$ vale $A[i+j] \geq i+j$).

A questo punto, un algoritmo può essere il seguente

```
Fix(A)
   return Fix-rec(A,1,n)

Fix-rec(A,p,r)
```