Giorno 2

1 Divide et Impera

L'approccio Divide et Impera consiste nel dividere un problema in sottoproblemi che sono versioni "più piccole" dello stesso. Si applicano tre passi:

1. **Divide**: Si divide il problema in sottoproblemi
2. **Impera**: Si applica ricorsivamente l'algoritmo per risolvere ognuno dei sottoproblemi
3. **Combina**: Si combinano le soluzioni dei sottoproblemi per generare la soluzione al problema iniziale

1.1 Merge Sort

Un esempio di algoritmo divide et impera è Merge Sort, un algoritmo di sorting ricorsivo.

```
1  MS(A, p, r):
2      if(r > p):
3          q = ⌊(p+r)/2⌋ ;
4          MS(A, p, q);
5          MS(A, q+1, r);
6          Merge(A, p, q, r);
```

```
1  Merge(A, p, q, r):
2      n1 = q - p + 1;
3      n2 = r - q;
4      L = nuovo array di lunghezza n1+1;
5      R = nuovo array di lunghezza n2+1;
6      for(i = 1 to n1): L[i] = A[p+i-1];
7      for(i = 1 to n2): R[i] = A[q+i];
8      L[n1 + 1] = +∞;
9      R[n2 + 1] = +∞;
10     i = 1; j = 1;
11     for(k = p to r):
12         if(L[i] ≥ R[j]):
13             A[k] = R[j]; j++;
14         else:
15             A[k] = L[i]; i++;
```

Costo di Merge: $\Theta(n)$

1.1.1 Invariante di ciclo di Merge

All'inizio della k-esima iterazione, $A[p...k-1]$ contiene, ordinati i $k-p$ elementi più piccoli di L e R.

1. **Inizializzazione**: Prima della prima iterazione si ha $k = p$, e zero elementi sono banalmente ordinati.
2. **Conservazione**: Ad ogni iterazione si inserisce in $A[k]$ o il primo elemento di $L[i, n_1]$ o il primo elemento di $R[j, n_2]$. Tali elementi sono rispettivamente il minimo di $L[i, n_1]$ e quello di $R[j, n_2]$ poiché i due array sono ordinati.
3. **Conclusione**: L'array A, alla fine dell'ultima iterazione, è ordinato poiché tutti gli elementi sono stati copiati tranne le sentinelle ($+\infty$).

1.2 Ricorrenze

Una ricorrenza è un'equazione o disequazione che definisce una funzione in termini del suo valore con input più piccoli. Ad esempio, la ricorrenza che definisce la complessità in tempo di Merge Sort è:

$$T(n) = \begin{cases} \Theta(1) & \text{Se } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{se } n > 1 \end{cases}$$

1.3 Metodi di risoluzione delle ricorrenze

1.3.1 Metodo di sostituzione

Si ipotizza una possibile soluzione e la si dimostra per induzione.

1.3.2 Metodo dell'albero di ricorsione

Si rappresentano i costi dei sottoproblemi in alberi: Si consideri ad esempio la ricorrenza di Merge Sort, che viene rappresentata come:

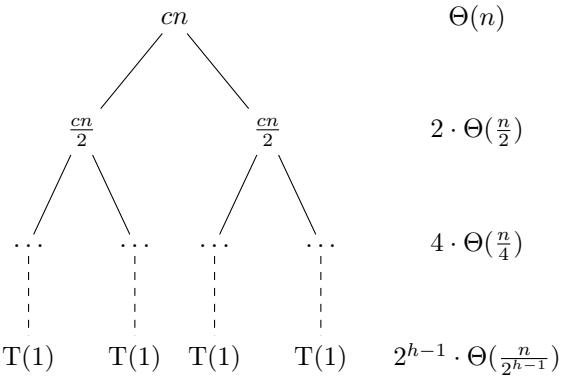
All' i-esimo livello il costo è

$$2^{i-1} \cdot \Theta\left(\frac{n}{2^{i-1}}\right) = \Theta(n)$$

Sia h l'altezza dell'albero: si ha che

$$\begin{aligned} T\left(\frac{n}{2^{h-1}}\right) = T(1) &\iff \frac{n}{2^{h-1}} = 1 \\ &\iff n = 2^{h-1} \\ &\iff h - 1 = \log_2 n \end{aligned}$$

Quindi l'altezza dell'albero è $h = \log_2 n + 1$. Per trovare il costo totale si sommano i costi ad ogni livello:



$$\sum_{i=0}^{\log_2 n+1} 2^{i-1} \cdot \Theta\left(\frac{n}{2^{i-1}}\right) = \sum_{i=0}^{\log_2 n+1} \Theta(n) = \Theta(n) \sum_{i=0}^{\log_2 n+1} 1 = \Theta(n) \cdot \log_2(n) + 1 = \Theta(n \log n)$$

1.3.3 Master Theorem

Teorema 1.1. Data una ricorrenza della forma $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$, $b \neq 0$:

1. Se $f(n) = O(n^{\log_b a - \varepsilon})$ per qualche $\varepsilon > 0$, allora $T(n) = \Theta(n^{\log_b a})$
2. Se $f(n) = \Theta(n^{\log_b a})$ allora $T(n) = \Theta(n^{\log_b a} \log n)$
3. - $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per qualche $\varepsilon > 0$
   - $af\left(\frac{n}{b}\right) \leq cf(n)$ definitivamente per qualche $c < 1$

   $\implies T(n) = \Theta(f(n))$

2 Quicksort

Quicksort suddivide ricorsivamente un array in due partizioni, una che contiene solo elementi maggiori di un certo elemento detto *pivot*, ed un altro che contiene solo gli elementi ad esso maggiori. Il pivot viene inserito tra le due partizioni, nella sua posizione finale.

```
1  QS(A, p, r):
2      if(r > p):
3          q = Partizion(A, p, r)
4          QS(A, p-1, q);
5          QS(A, q+1, r);
```

```
1  Partition(A, p, r):
2      x = A[r]
3      i = p-1; j = p;
4      while(j<r):
5          if(A[j] ≤ x):
6              i++;
7              scambia A[i], A[j];
8          j++;
9      scambia A[i+1], A[r]
10     return i+1;
```

Costo di Partition: $\Theta(n)$

2.0.1 Invariante di ciclo di Partition

All'inizio della j esima iterazione:

1. Il sottoarray $A[p..i]$ contiene solo elementi minori di x;
2. Il sottoarray $A[i+1..j-1]$ contiene solo elementi maggiori di x;
3. $A[r] = x$.

1. **Inizializzazione**: Prima della prima iterazione i due sottoarray sono vuoti, quindi le proprietà sono vacuamente soddisfatte;
2. **Conservazione**: Ad ogni iterazione, se $A[j]$ è minore di x si incrementa i e si scambiano $A[i] ed A[j]$ (proprietà 1), mentre se $A[j]$ è maggiore di x, si incrementa semplicemente j (proprietà 2).
3. **Conclusione**: Alla fine del ciclo, $A[p..i]$ e $A[i+1..r-1]$ sono le due partizioni, e $A[r]$ vale ancora x.

2.1 Complessità di Quicksort

La complessità di QS varia tra il caso ottimo, il caso pessimo ed il caso medio.

2.1.1 Caso ottimo

Al caso ottimo l'albero di ricorsione è bilanciato (il pivot è sempre il valor medio dell'array). In questo caso la ricorrenza che caratterizza $T(n)$ è:

$$T(n) \leq \begin{cases} \Theta(1) & \text{Se } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{se } n > 1 \end{cases}$$

La cui soluzione sappiamo essere (per il teorema dell'esperto) $T(n) = n \log n$

2.1.2 Caso pessimo

Al caso pessimo l'albero di ricorsione è molto sbilanciato, poiché il pivot è sempre maggiore o minore di tutti gli elementi (quindi una delle partizioni è sempre vuota). Si ha che:

$$T(n) = \begin{cases} \Theta(1) & \text{Se } n = 1 \\ T(n-1) + T(0) + \Theta(n) & \text{se } n > 1 \end{cases}$$

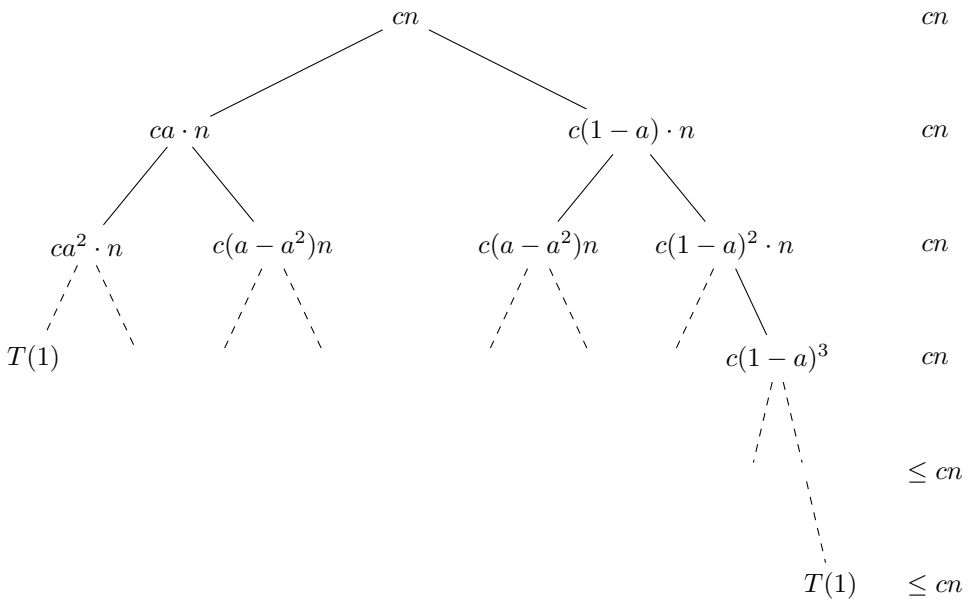
L'albero di ricorsione ha profondità n, ed il costo ad ogni livello è $\Theta(n)$, perciò si ha un costo totale di $n \cdot \Theta(n) = \Theta(n^2)$

2.1.3 Ripartizione bilanciata

Nel caso di ripartizione bilanciata, dato $a < 1$, si ha:

$$T(n) \leq \begin{cases} \Theta(1) & \text{Se } n = 1 \\ T(a \cdot n) + T((1-a) \cdot n) + \Theta(n) & \text{se } n > 1 \end{cases}$$

Quindi l'albero di ricorsione, assumendo ad esempio $a > \frac{1}{2}$, ha questa forma:



(e ok do go on)