1° lezione di algoritmi

- [x] ~~Revisionata~~

Problemi di ottimizzazione

Un problema di ottimizzazione è un tipo di problema matematico o computazionale in cui devi trovare la migliore soluzione possibile tra un insieme di possibili soluzioni, solitamente in base a un certo criterio di ottimizzazione. L'obiettivo è massimizzare o minimizzare una funzione obiettivo, detta anche funzione di costo o di utilità, a seconda della natura del problema. Questa funzione di costo rappresenta una misura di quanto "buona" o "cattiva" sia una soluzione.

Come riconoscere un problema di ottimizzazione?

I problemi di ottimizzazione hanno solitamente molteplici soluzioni possibili, e il tuo obiettivo è trovare la migliore di queste soluzioni in base alla metrica definita dalla funzione obiettivo.

Algoritmi di ordinamento e loro complessità

Selection Sort

Il selection sort o algoritmo di ordinamento per selezione trova il minimo a ogni iterazione e lo sostituisce con l'elemento che sta analizzando. **Ha complessità** $O(N^2)$

```
void selection_sort(int vector[], int size ){
    for(int i=0; i < size - 1 ; i++){
        for(int j=i+1; i < size; i++){
            if(vector[i] < vector[j]{
                swap(vector[i], vector[j]) //find min element and replace current elemet to min
        }
    }
}

void swap(int a, int b){
  int temp=b
    a = b;
    b = temp;
}
```

Un esempio di un ordinamento:

[8] 4 6 [1] 2 7 5 3

1 [4] 6 8 [2] 7 5 3

1 2 [6] 8 4 7 5 [3]

1 2 3 [8] [4] 7 5 6

1 2 3 4 [8] 7 [5] 6

1 2 3 4 5 [7] 8 [6]

1 2 3 4 5 6 [8] [7]

Inserction Sort

L'Insertion Sort è un algoritmo di ordinamento semplice ma efficiente per piccoli insiemi di dati. Funziona costruendo una sequenza ordinata di elementi, uno alla volta. Ad **ogni iterazione, l'algoritmo prende un elemento dalla lista non ordinata e lo inserisce nella posizione corretta all'interno della parte già ordinata della lista.** Questo processo continua finché tutti gli elementi non sono stati inseriti nella sequenza ordinata.

La complessità di quest'algoritmo è $O(N^2)$

```
void insertion_sort(int vector[], int size){
    for (int index = 0; index < size; index++) {
        int key = vector[index];
        int position = index;
        while (position > 0 && vector[position - 1] > key){
            vector[position] = vector[position - 1];
            position--;
        }
        vector[position] = key;
    }
}
```

La caratteristica più importante di questo algoritmo è che è adattivo ciò vuol dire che si adatta a l'input che gli viene passato.

Il **caso migliore** quando l'algoritmo è già ordinato.

Il **caso peggiore** si verifica quando l'array è ordinato in ordine inverso. In questo scenario, ogni elemento deve essere spostato alla posizione iniziale dell'array, risultando in un numero massimo di confronti e scambi. La complessità temporale media dell'Insertion Sort è O(n^2), rendendolo efficiente per piccoli dataset ma meno pratico per grandi insiemi di dati.

[8] 4 6 1 2 7 5 3

[4 8] 6 1 2 7 5 3

[4 6 8] 1 2 7 5 3

[1 4 6 8] 2 7 5 3

[1 2 4 6 8] 7 5 3

[1 2 4 6 7 8] 5 3

[1 2 4 5 6 7 8] 3