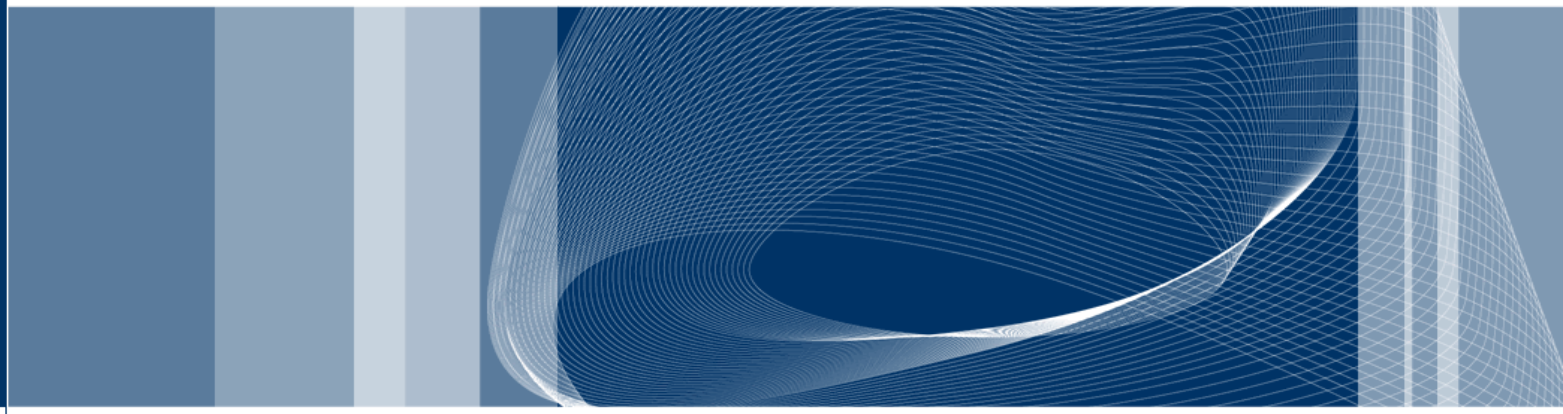
POLITECNICO DI MILANO



Esercizi di riepilogo

Algoritmi e Calcolo Parallelo

Daniele Loiacono

Esercizio 1

- Si consideri un problema di ottimizzazione in cui ogni soluzione s è un array di **n** float (compresi tra 0 e 1). Si ha a disposizione una funzione float f(float *s) che fornisce una valutazione della funzione s (maggiore è il valore ritornato da f, migliore è la soluzione s). Si supponga inoltre di avere a dispozione una funzione `bool intorno(float *s_in, float *s_out, int i)` che restituisce true e copia in s_out l'i-esima (con i>=0) soluzione nell'intorno di s_in (la funzione ritorna false e non modifca s_out se s_in ha meno di i soluzioni nell'intorno).
- Si implementi, utilizzando i pthread, una versione parallela della **variante random restart** dell'algoritmo **hill climber**

Esercizio 1 – Hill Climber


```
genera una soluzione iniziale s di costo z(s)
while (not CRITERIO_TERMINAZIONE) {
   genera l'intorno N(s)
   trova la migliore soluzione s' ∈ N(s) rispetto a z(·)
   if (z(s') < z (s)) {
      s = s'
   else
      TERMINA_RICERCA
   }
}
s*=s /*salva la migliore soluzione*/
```

- **Variante Random-restart**
  - Ripete diverse volte l'algoritmo base partendo da diverse soluzioni generate casualmente
  - In alcuni problemi può essere sufficiente per evitare ottimi locali

Esercizio 2

- Applicare l'algoritmo di Huffman per costruire l'albero di codifica/decodifica ottimo per il seguente messaggio: CTTGCACTCTGTGAT
- Utilizzare poi il codice ottenuto per decodificare il seguente messaggio: 100011110110

Esercizio 2 - Soluzione

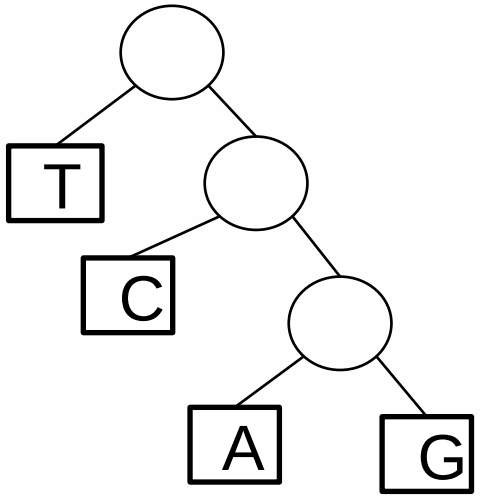
Codifica

T:0

C:10

A:110

G:111



100011110110 → CTTGCA

Esercizio 3

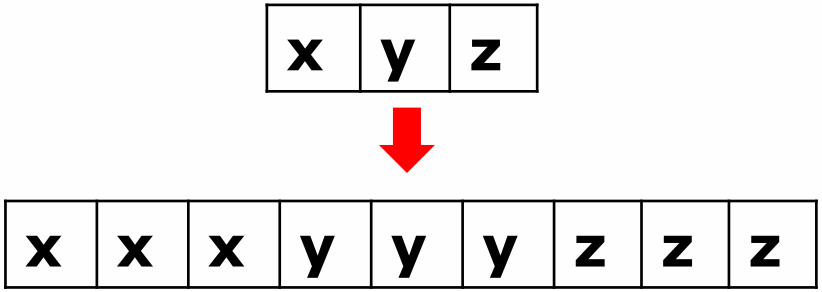
- Dato un array di interi v, definiamo v’ un array tale che

  v’[i] = v[i] – v[i-1] per ogni i>=1 e v’[0] = 0

- Implementare in MPI il calcolo del vettore v’

Esercizio 4

- Implementare in CUDA un algoritmo parallelo per replicare gli elementi di un vettore come segue:



- Nell'implementazione tenere in considerazione le problematiche di accesso alla memoria globale

Esercizio 5

- Parallelizzare con OpenMP il seguente codice:

```
float b,x,sum=0;
cin >> b;
x=1;
for (int i=0; i<N; i++){
    sum += x;
    x *= b;
}
cout << "Sum = " << sum << endl;
```

- Se necessario riprogettare l'algoritmo

Esercizio 6

- Implementare utilizzando i pthread tournament selection di cui si riporta una implementazione seriale di seguito:

```
void binary_ts (float *f, int *S, int n){
    for (int i=0; i<n; i++){
            a = rand(1,n);
            b = rand (1,n-1);
            if (b>=a) b++;
            if (f[a]>f[b])
                    S[i] = a;
            else
                    S[i] = b;
    }
}
```

Esercizio 6bis

- Risolvere il precedente esercizio con CUDA
- Fare un’analisi critica della soluzione proposta