UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di laurea magistrale in ingegneria informatica

Corso di architetture e programmazione dei sistemi di elaborazione

Progetto A.A. 2018/2019

Algoritmo “Product Quantization for Nearest Neighbor Search” in linguaggio assembly x86-32+SSE e x86-64+AVX

Gruppo9

Gianpaolo Cascardo	189088
Davide Galati	189222
Maria Chiara Nicoletti	166773

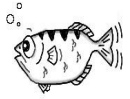
Docente

Fabrizio Angiulli

Scopo e specifiche di progetto

Visual Studio Code: code editor, avanzato e programmabile, sviluppato e fornito da Microsoft per i sistemi Linux, Windows e macOS tramite il framework di sviluppo Electron

GDB e CGDB: gdb è un debugger con supporto a diversi linguaggi, tra cui *C* ed *Assembly*. Nel progetto si è abusato della sua versione scritta tramite il framework *ncurses* che ne consente un'interazione grafica minimale tramite terminale



WORKFLOW

Proiezione dello spazio iniziale del dataset in sottospazi a dimensione ridotta e processarne ognuno separatamente

Il lavoro è stato suddiviso in due fasi: stesura di una prima versione funzionante, ottimizzazione dei punti critici.

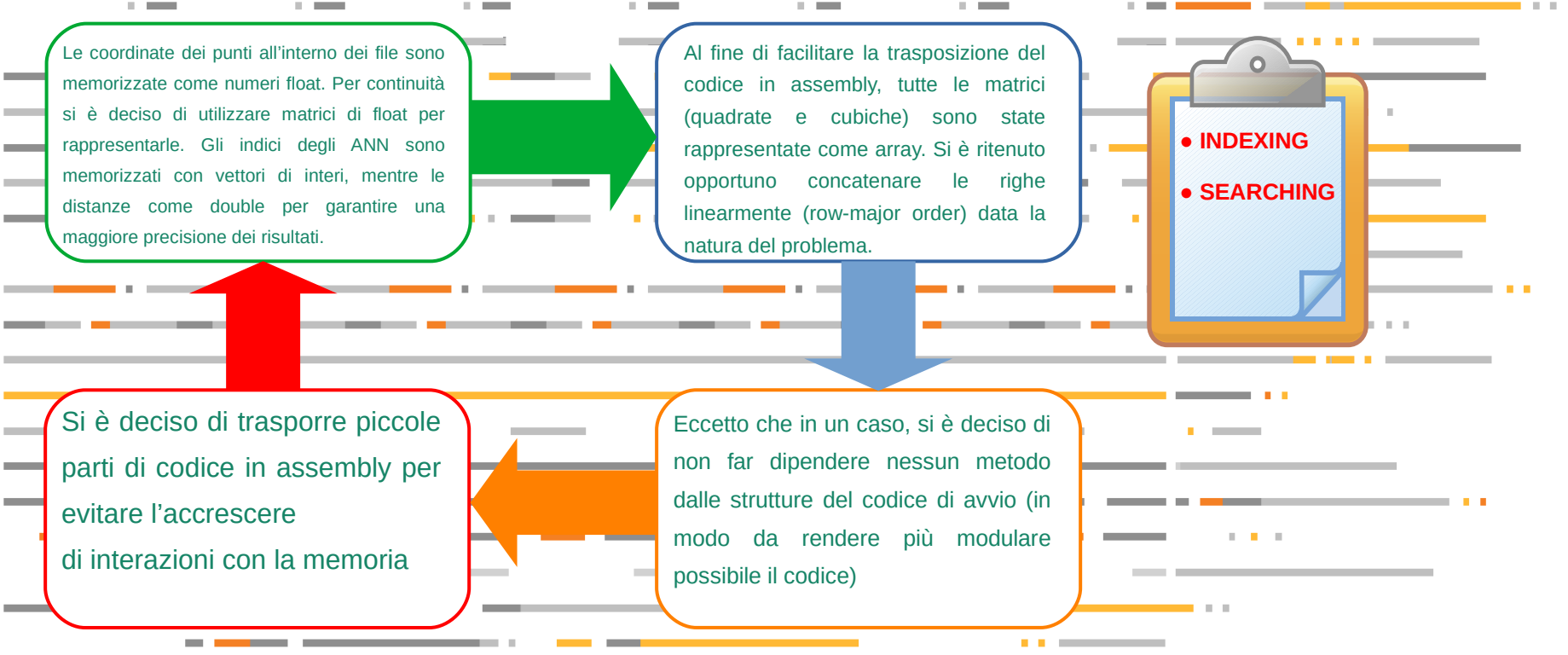
Realizzare una versione ottimizzata, attraverso l'utilizzo delle estensioni SSE ed AVX, di un algortimo di "Product Quantization for Nearest Neighbor Search"

1. STUDIO DELLE TECNOLOGIE E DEL PROBLEMA
2. IMPLEMENTAZIONE AD ALTO LIVELLO
3. IMPLEMENTAZIONE A BASSO LIVELLO
4. VALUTAZIONE DELLE PRESTAZIONI E POSSIBILI MIGLIORAMENTI

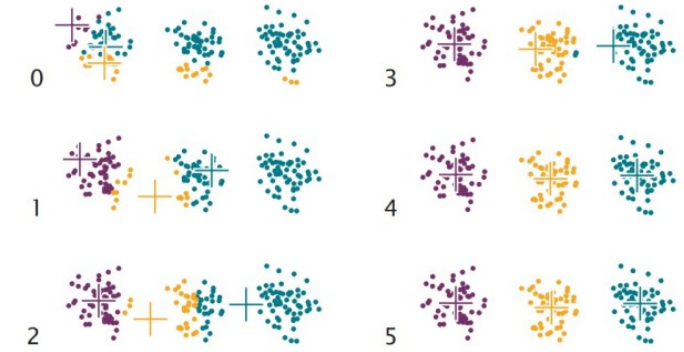
~MJ

Progettazione e implementazione

Il lavoro progettuale è iniziato con la rappresentazione in memoria del dataset. Si è scelto infatti di memorizzare il singolo punto in un array d-dimensionale e l'intero set come una concatenazione di n punti consecutivi. Similmente sono stati rappresentati i punti del codebook (centroidi) e quelli del queryset (le interrogazioni).



Indexing



DIST_E_2 (D,X,XI,Y,YI,RIS)

```
float somma=0
    differenza=0;
    for(register int i=0;i<D;i++){
            differenza=X[(XI)+i]-Y[(YI)+i];
            differenza*=differenza;
            somma+=differenza;
    }
RIS=somma;
```

Nel caso di ricerca non esaustiva: dataset e codebook sono ridotti

.

void init_codebook(int d, int n, float* dataset, int k, float* codebook)

void k_means(int d, int m, float eps, int tmin, int tmax, int k, float* codebook, int n, float* dataset, int* map)

void pq(int d, int m, int k, float* codebook, int n, float* dataset, int* map)

KMEANS_STEP(d,m,n,dataset,map,k,codebook)

void nuovicentroidi (int d,int m,int n, float* dataset,int* map,int k,float* codebook)

double obiettivo(int d, int m, int n, float* dataset, int *map,float* codebook)

int mindist(int d, int dstar, int mi, float *dataset, int di, int k, float* codebook)

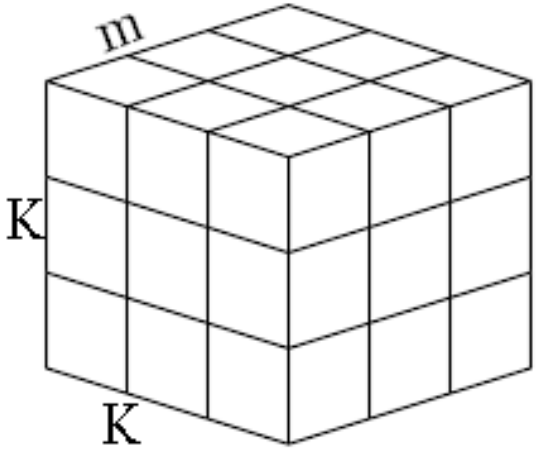
Searching

La ricerca avviene minimizzando la somma delle distanze al quadrato:

- In caso di quella simmetrica questo confronto si basa sulla distanza tra i centroidi associati al queryset e quelli associati al dataset
- Quella asimmetrica, invece, la calcola tra i punti del queryset ed i centroidi associati al dataset

```
void ANNSDC (int d, int m, int k, float* codebook, int K, int*ANN, double* ANN_values, int n, int*map, int nq, float*qs)
void ANNADC (int d, int m, int k, float* codebook, int K, int*ANN, double* ANN_values, int n, int*map, int nq, float*qs)
void ADC (int d, int k, int m, double* distanze, int K, int*ANN, double* ANN_values,int n, int* map, int ix, float *qs)
void SDC (int d, int k, int m, int nrd, double* distanze, int K, int*ANN,double* ANN_values, int n, int* map, int ix, int *qx)
void mergeSort (double* values, int* indices, int start, int end, int offset)
```

Strutture dati

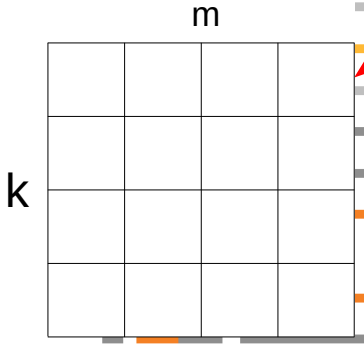


Precalcolo delle differenze:

Nel caso della ricerca SDC si è utilizzata una matrice cubica k×k×m. All'indice corrisponde la distanza tra il centroide i e il centroide j nel sottospazio w

nel caso della ricerca ADC si è utilizzata una matrice quadrata
k ×m precalcolata per ogni punto del queryset. All'indice (i,w) corrisponde la distanza tra il punto e il centroide i nel sottospazio w

in caso di ricerca non esaustiva si utilizza, una delle due tecniche sopracitate basandosi sul confronto tra i residui dei punti. La ricerca si limita ad un sottoinsieme di centroidi considerati più vicini al punto del queryset

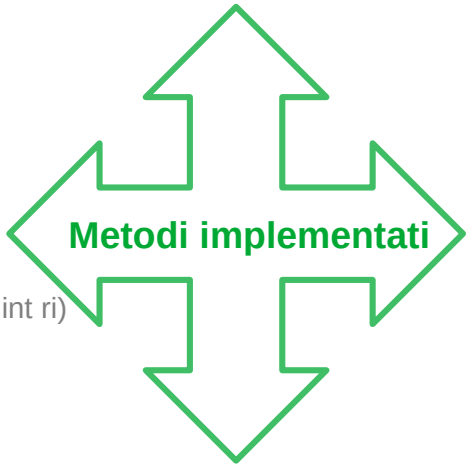


Assembly

DIST_E_2(D,X,XI,Y,YI,RIS) tradotto nella procedura
float dist_2_asm(int d,float *x,int xi,float *y,int yi)

DIFFVF(D,X,XI,Y,YI,RES,RI) tradotto nella procedura

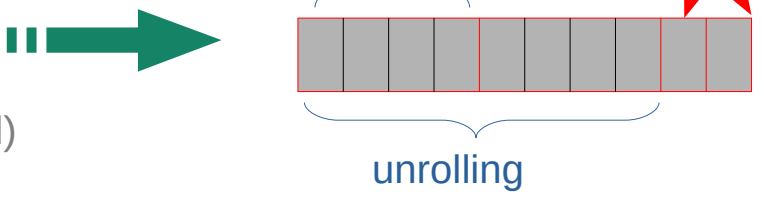
void diffvf_asm(int d,float *x,int xi,float *y,int yi,float * res,int ri)



INIT_ARRAY(D,A,AI,IV) tradotto nella procedura
void azzera_array(int d,float*v)

COPYV(D,DEST,DESTI,SRC,SRCI) tradotto nella procedura
void copyv_asm(int d,float* dest,int desti,float *src,int srci)

Tecniche ottimizzazione usate
- Code vectorization
- Loop unrolling (mod)



Implementazione assembly

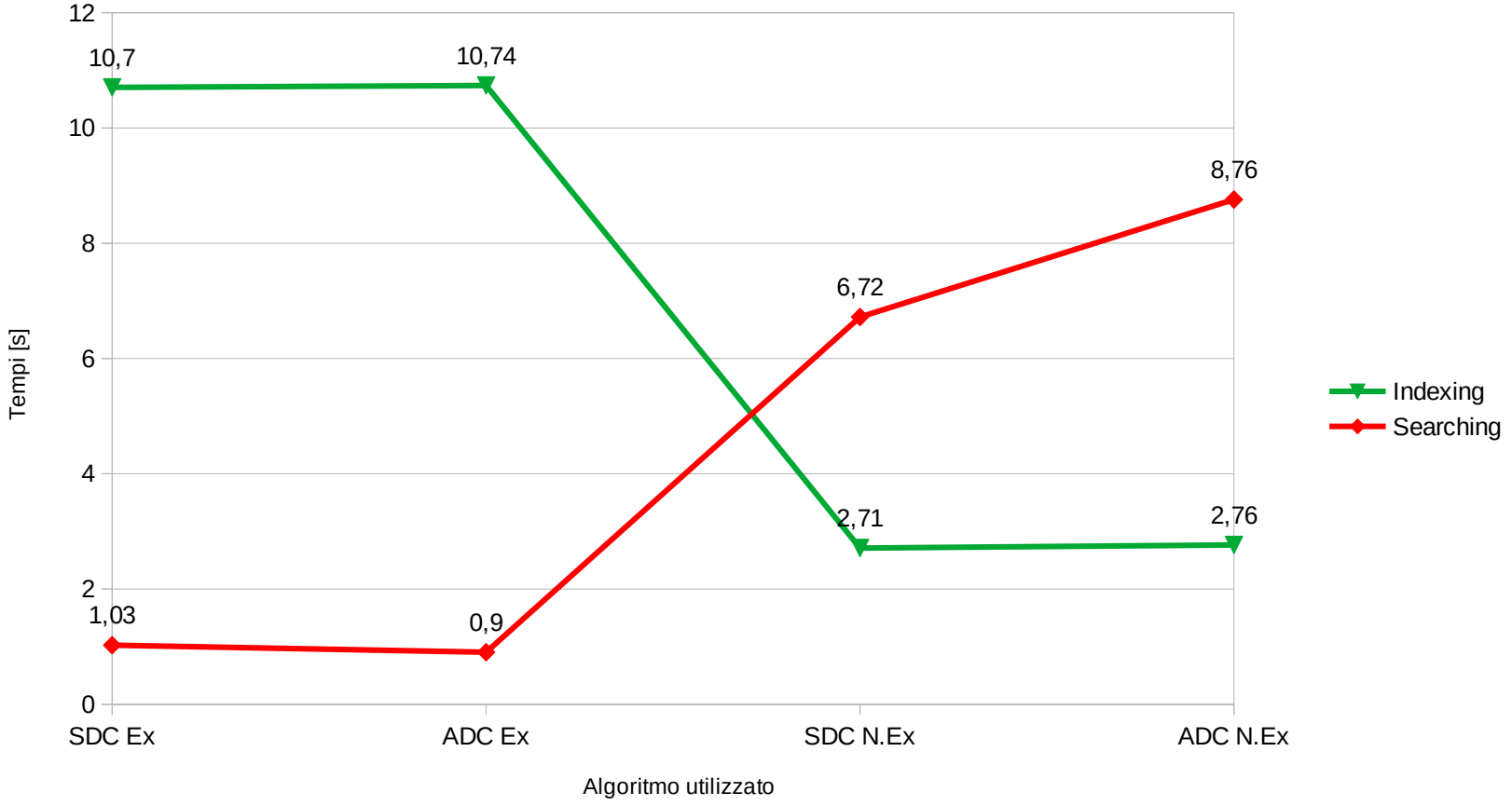
```
dist_2_asm:
    ; ... inizializzazioni ...
    mov   esi, ebx ;esi=d
    sub   esi, 4*R ;esi=d-p*r
    inc   esi      ;esi=d-p*r+1

LOOPRDIST:
    cmp edi, esi
    jge LOOPSDIST

    dist_2_step
    ; ...
    ; copia r volte
    ; ...
    dist_2_step

    jmp     LOOPRDIST

LOOPSDIST:
    cmp edi, ebx
    jge ENDDIST
    mov      edx, [ebp+16]
    add      edx, edi
    movss   xmm1, [eax+edx*4]
    mov      edx, [ebp+24]
    add      edx, edi
    subss  xmm1, [ecx+edx*4]
    mulss   xmm1,xmm1
    addss   xmm0,xmm1
    inc edi
    jmp LOOPSDIST

ENDDIST:
    haddps xmm0, xmm0
    haddps xmm0, xmm0
    movd [ebp-4], xmm0
    fld dword [ebp-4]
```

```
%macro dist_2_step 0
    mov      edx, [ebp+16]
    add      edx, edi
    movups   xmm1, [eax+edx*4]
    mov      edx, [ebp+24]
    add      edx, edi
    subps    xmm1, [ecx+edx*4]
    mulps    xmm1,xmm1
    addps    xmm0,xmm1
    add      edi, 4
%endmacro
```

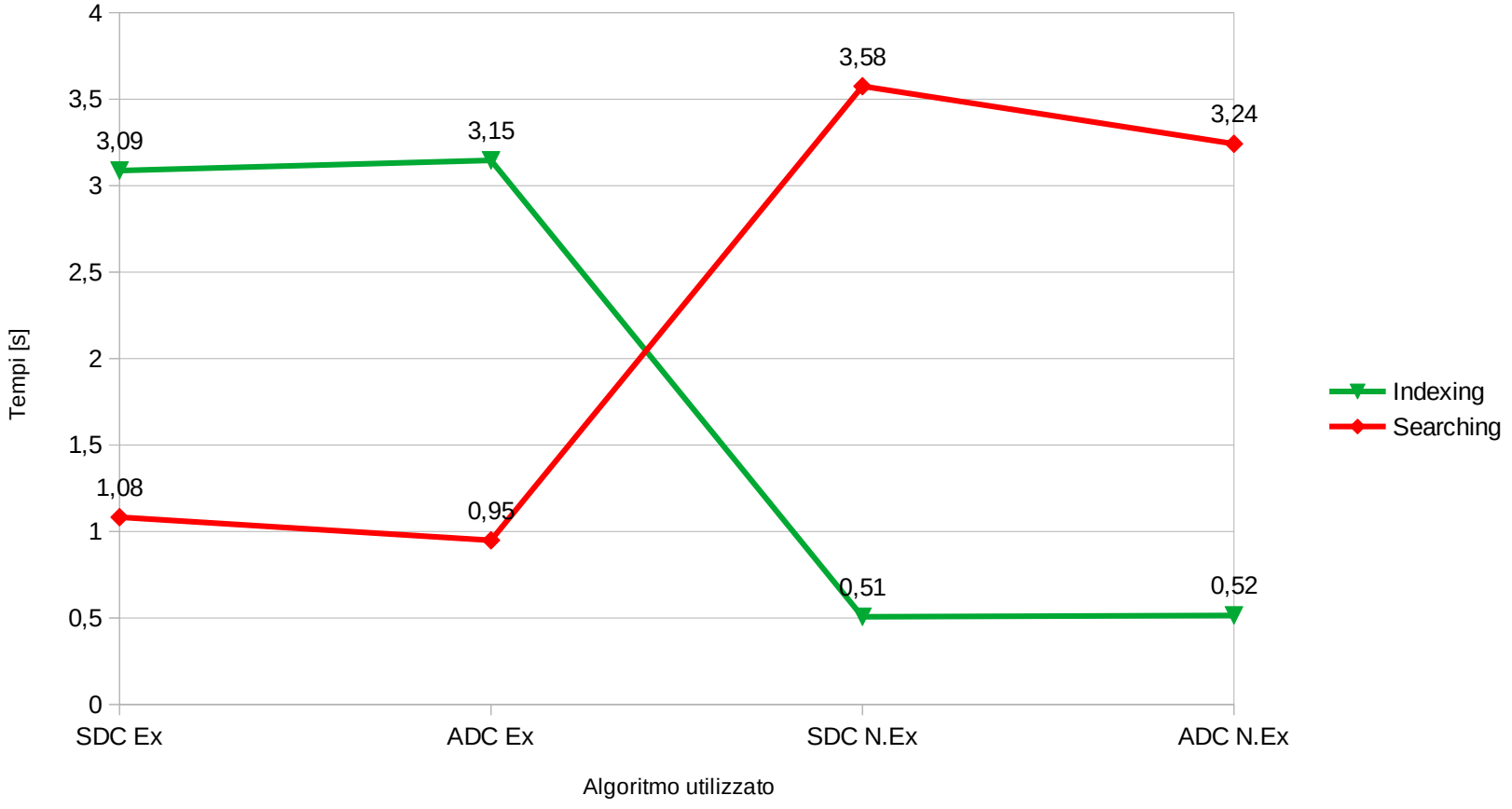
Le macro aiutano a rendere il codice più modulare

Ne viene meno uno dei vantaggi dell'unrolling

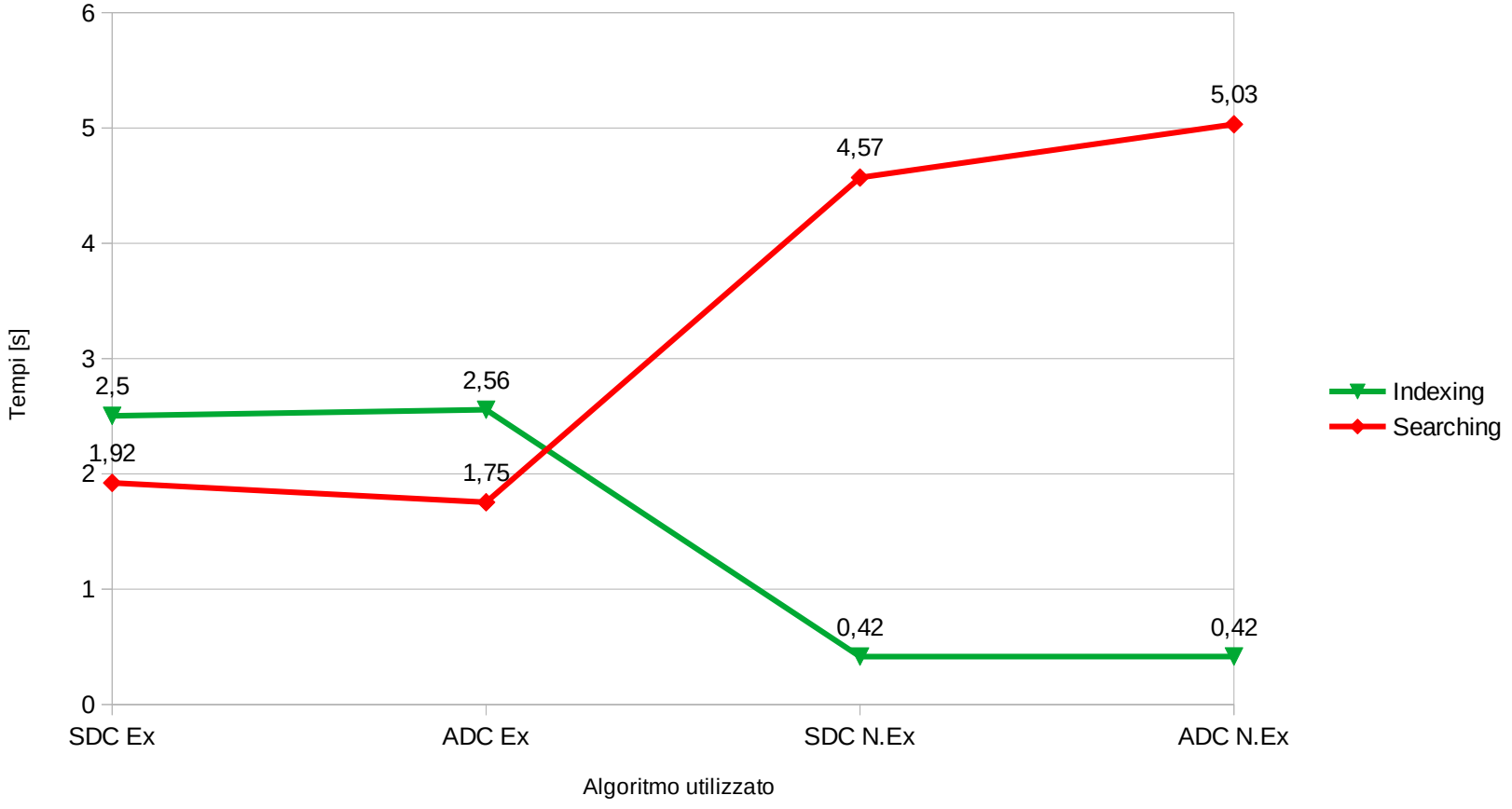
Tempi e prestazioni: C



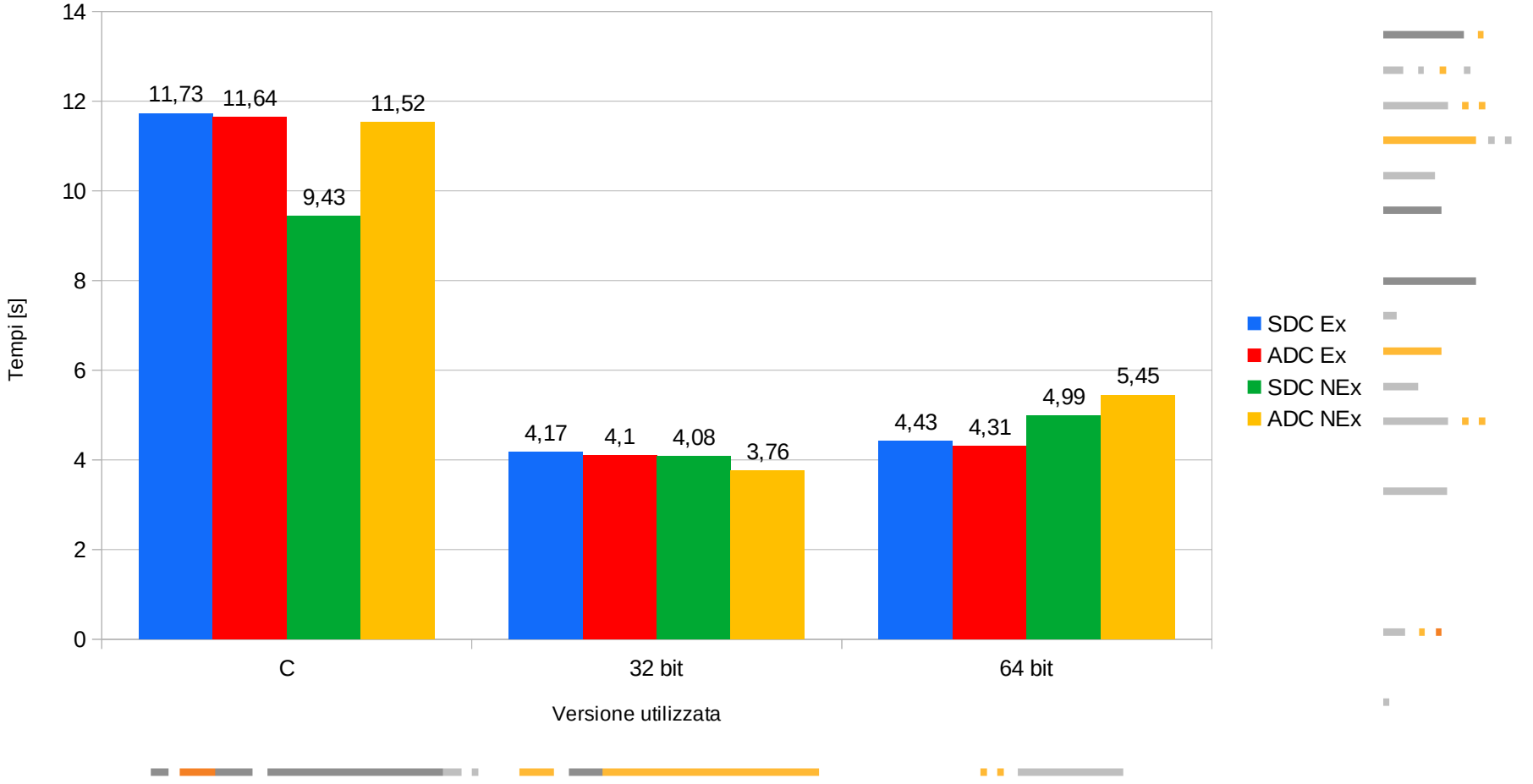
Tempi e prestazioni: ASM 32bit



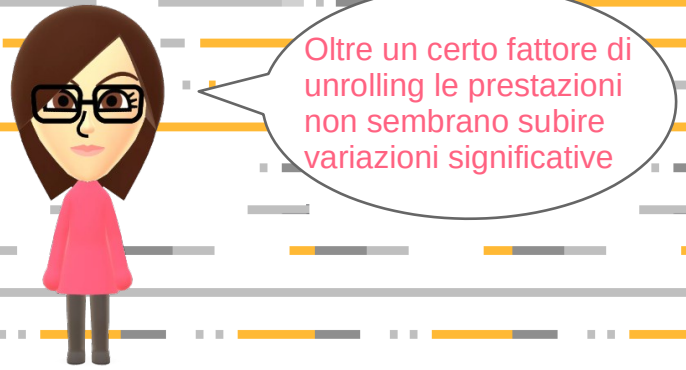
Tempi e prestazioni: ASM 64bit



Tempi e prestazioni: Totali

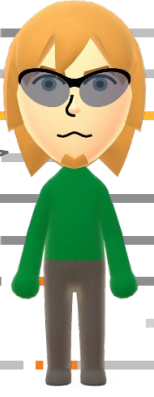


Conclusioni



Oltre un certo fattore di unrolling le prestazioni non sembrano subire variazioni significative

Non tutte le traduzioni in assembly hanno influenzato le prestazioni in fase di esecuzione



Valutare delle implementazioni a basso livello per ridurre i tempi di esecuzione di algoritmi strutturalmente più complessi

0x54 0x68 0x65 0x20 0x45 0x4e 0x44