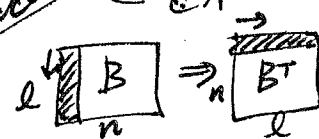
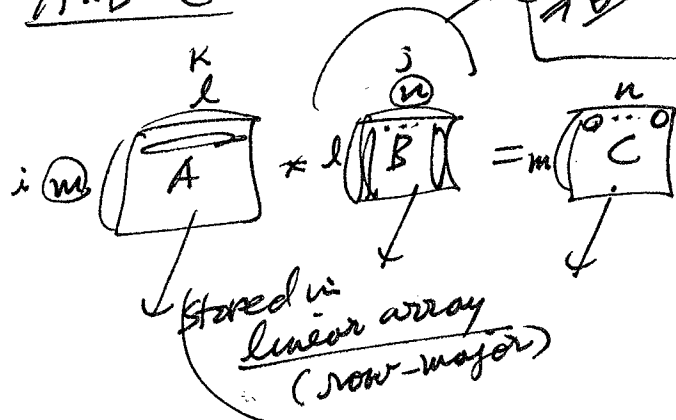


20/8/2008

Listing 3.1 matrix mult

$$A * B = C$$



row-major layout (C++)
 $A[i,j] = (i * \#cols) + j$

naive mult

```

for (i=0; i<m; i++)
  for (j=0; j<n; j++)
    Acc=0;
    for (k=0; k<l; k++)
      Acc += A[i][k] * B[k][j]; // dot product
    C[i][j] = Acc;
  
```

Diagram illustrating the naive multiplication loop. The inner loop calculates the dot product $A[i][k] * B[k][j]$. The result is stored in $C[i][j]$. The diagram shows the row-major layout of C and the row-major layout of B .

transpose $B \rightarrow B^T$ and mult.

```

for (k=0; k<l; k++)
  for (j=0; j<n; j++)
    B^T[j][k] = B[k][j];
  
```

Diagram illustrating the transpose operation. Matrix B is $l \times n$, and B^T is $n \times l$. The transpose operation is shown as $B^T[j][k] = B[k][j]$.

↓
- transposed-mult

```
for (i=0; i<m; i++)
```

```
  for (j=0; j<n; j++)
```

```
    Acc=0;
```

```
    for (k=0; k<l; k++)
```

```
      Acc += A[i*l+k] * BT[j*l+k];
```

$A[i \cdot l + k]$

$B^T[j \cdot l + k]$

$\equiv B[k \cdot l + j]$ (col-major access)
(same entry) \downarrow more penalty

(less cache miss penalty)

(row-major access)

concept

$$\begin{matrix} m & l \\ \boxed{A} & * \begin{matrix} l & n \\ \boxed{B} \end{matrix} = \begin{matrix} m & n \\ \boxed{C} \end{matrix} \end{matrix}$$

⇒

$$\begin{matrix} m & l \\ \boxed{A} & * \begin{matrix} l & n \\ \boxed{B^T} \end{matrix} = \begin{matrix} m & n \\ \boxed{C} \end{matrix} \end{matrix}$$

performance

- on i7, $m=n=l=2^{13}$

vs. [naive mult — 5559.46 sec

transposed-mult — 496.66 sec

transpose : 0.752 sec
mult : 497.91 sec