Data Structure I: Tutorial 5

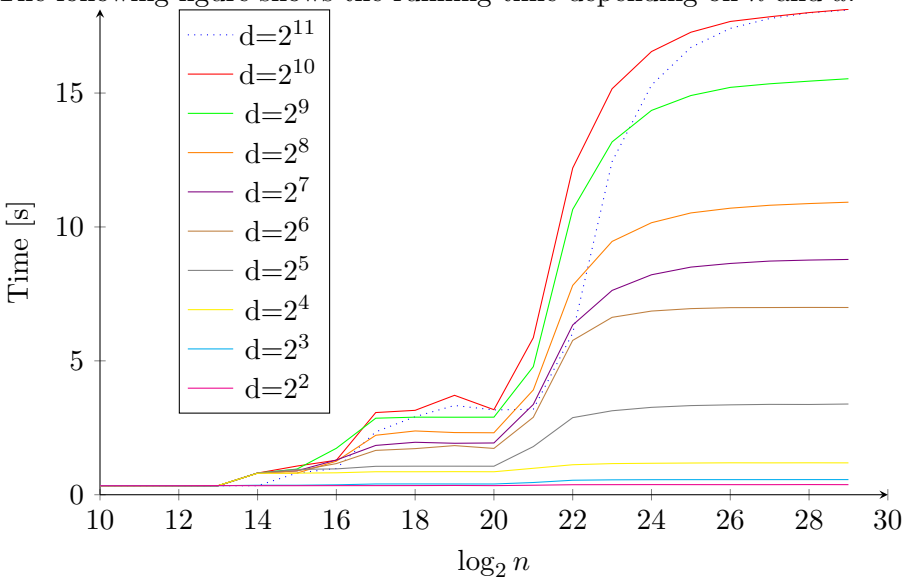
Example of sizes and speeds of different types of memory:

	size	speed
L1 cache	32 KB	223 GB/s
L2 cache	256 KB	96 GB/s
L3 cache	8 MB	62 GB/s
RAM	32 GB	23 GB/s
SDD	112 GB	448 MB/s
HDD	2 TB	112 MB/s
Internet	∞	10 MB/s

Consider the following trivial program:

```
# Initialize an array A of 32-bit integers of length n
for (i=0; i+d<n; i+=d) do
    A[i] = i+d # Create a loop using every d-th position
A[i]=0 # Close the loop
for (j=0; j< 2^28; j++) do
    i=A[i] # Repeatedly walk on the loop
```

The following figure shows the running time depending on n and d.



Exercise 1 *Estimate the number of cache misses for the following trivial algorithm transposing a matrix.*

```
for i ← 2 to k do
    for j ← i + 1 to k do
        Swap(A_ij, A_ji)
```

Exercise 2 *The following algorithm splits a matrix of size $k \times k$ into submatrices of size $z \times z$ assuming z divides n. Estimate the number of cache misses.*

```
  # We split the matrix A into submatrices of size z × z
1 for (i = 0; i < k; i+ = z) do
2     for (j = i; j < k; j+ = z) do
          # We transpose the submatrix starting on position (i, j)
3         for (ii = i; ii < min(k, i + z); ii + +) do
4             for (jj = max(j, ii + 1); jj < min(k, j + z); jj + +) do
5                 Swap(A_ii,jj, A_jj,ii)
```

Exercise 3 *In the assignment, implement the following recursive algorithm for matrix transposition.*

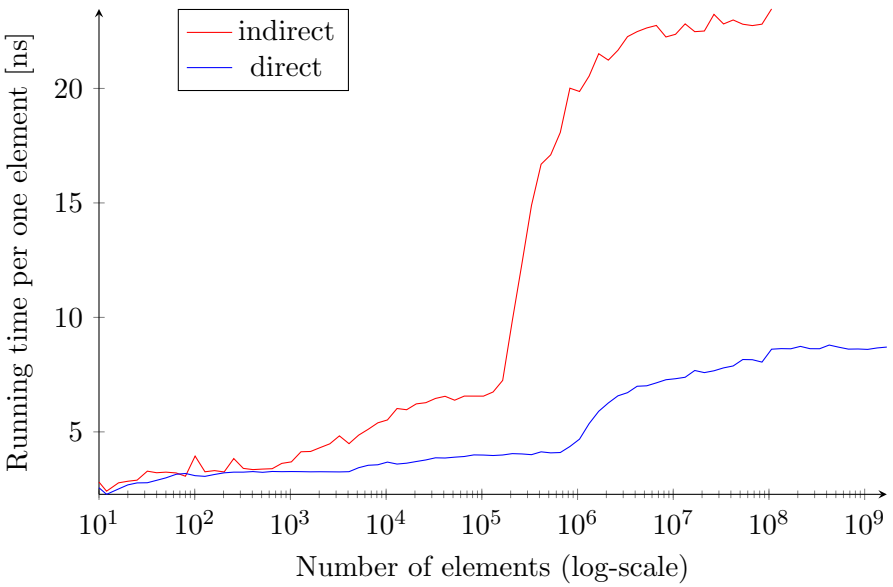
```
 1 Procedure transpose_on_diagonal(A)
 2     if matrix A is small then
 3         Transpose matrix A using the trivial approach
 4     else
 5         A11, A12, A21, A22 ← coordinates of submatrices
 6         transpose_on_diagonal(A11)
 7         transpose_on_diagonal(A22)
 8         transpose_and_swap(A12, A21)

 9 Procedure transpose_and_swap(A, B)
10     if matrices A and B are small then
11         Swap and transpose matrices A and B using the trivial approach
12     else
13         A11, A12, A21, A22, B11, B12, B21, B22 ← coordinates of submatrices
14         transpose_and_swap(A11, B11)
15         transpose_and_swap(A12, B21)
16         transpose_and_swap(A21, B12)
17         transpose_and_swap(A22, B22)
```

Exercise 4 *In the recursive algorithm, describe splitting of a matrix which size is not a power of two.*

Exercise 5 *Why the tall-cache assumption is needed in the proof of I/O complexity on matrix transposition? What happens when this assumption is not satisfied?*

The following compares the running time of building a binary heap depending on its size. The blue line shows the running time when all data are stored in heap's array. For the red line, the heap's array contains pointers to data and memory is allocated for every element separately.


indirect
direct
Running time per one element [ns]
20
15
10
5
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Number of elements (log-scale)