Divide & Conquer

สมชาย ประสิทธิ์จูตระกูล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

(๑๔ มีนาคม ๒๕๔๘)

หัวข้อ : Divide & Conquer

- Sorting
  - Mergesort
  - Quicksort
- Searching
  - Selection
  - Binary search
  - BSP

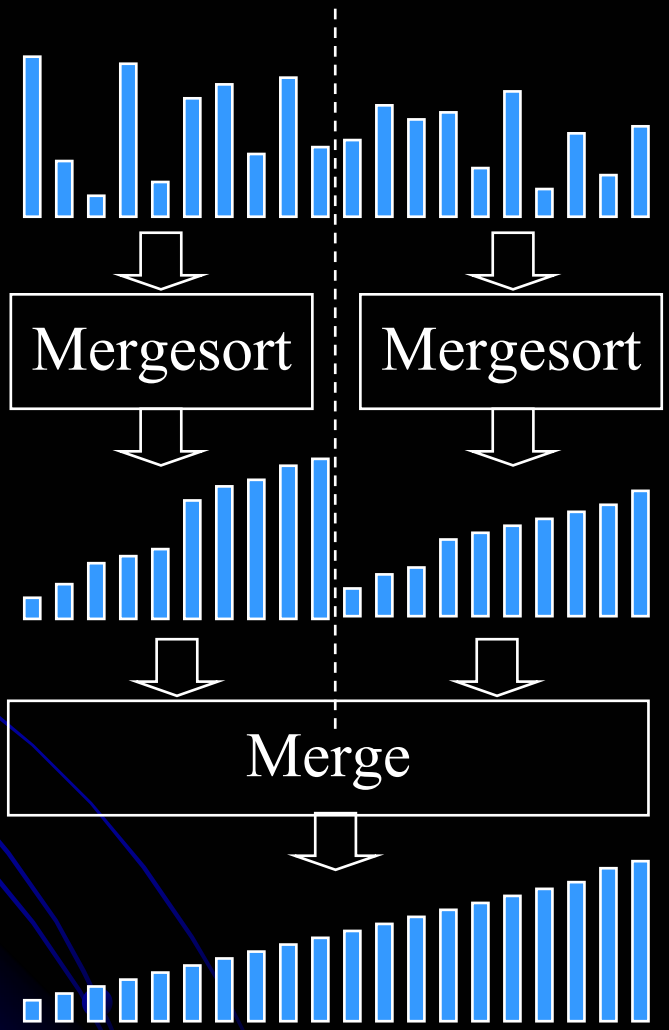
somchaip@chula.ac.th

Divide and Conquer

```
DQ( P ) {

  if ( P is trivial ) return Solve( P )

  Divide P into P1, P2, …, Pk

  for ( i = 1 to k )
    Si = DQ( Pi )

  S = Combine( S1, S2, …, Sk )

  return S
}
```


somchaip@chula.ac.th

Mergesort



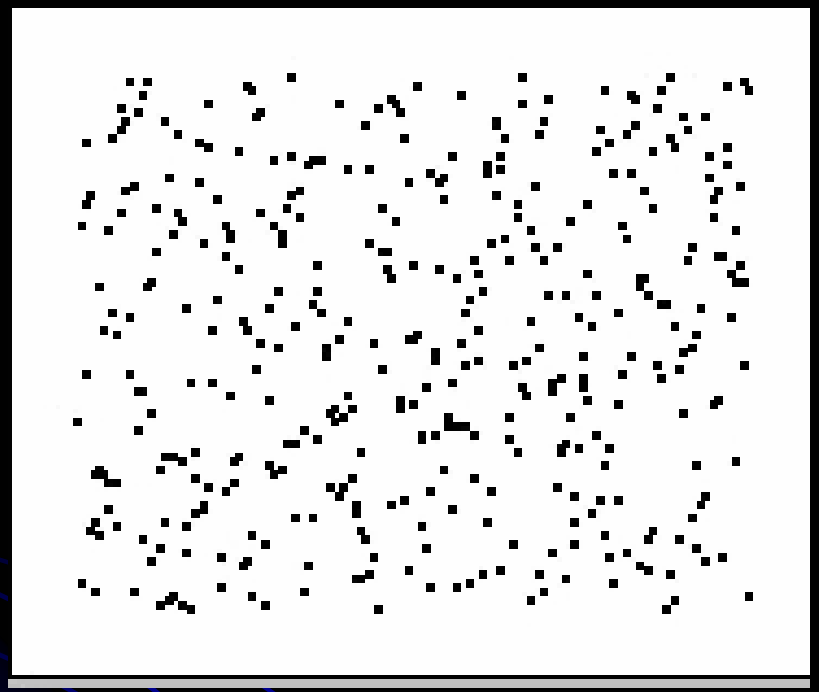
$2T(n/2)$

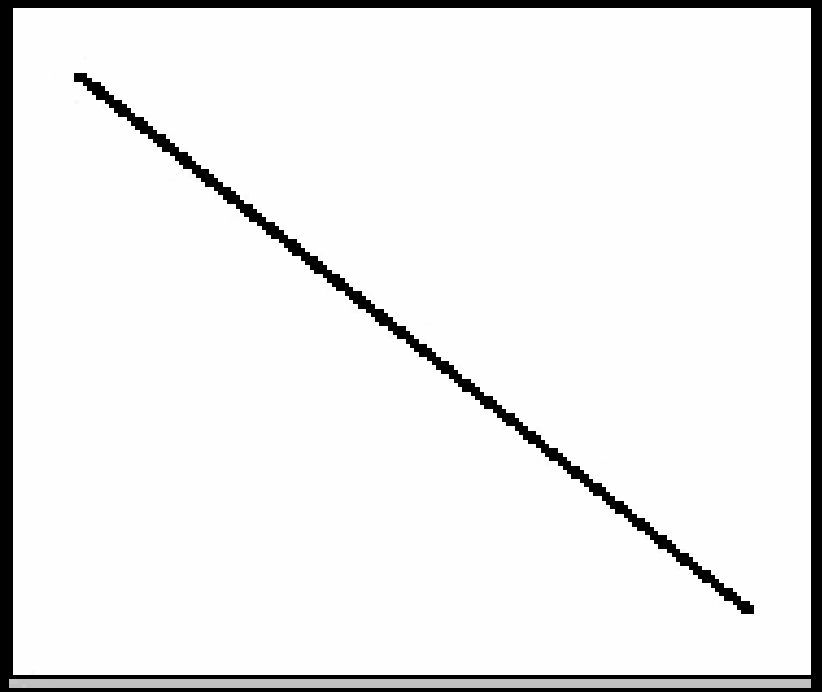
$$T(n) = 2T(n/2) + \Theta(n)$$

$\Theta(n)$

somchaip@chula.ac.th

Mergesort





somchaip@chula.ac.th

Mergesort : Analysis

- $T(n) = 2T(n/2) + \Theta(n)$
- Master method : $a = 2, b = 2, f(n) = \Theta(n)$
- $c = \log_b a = \log_2 2 = 1$
- $n^c = n^1 \quad f(n) = \Theta(n^c) = \Theta(n)$
- $T(n) = \Theta(n^c \log n) = \Theta(n \log n)$

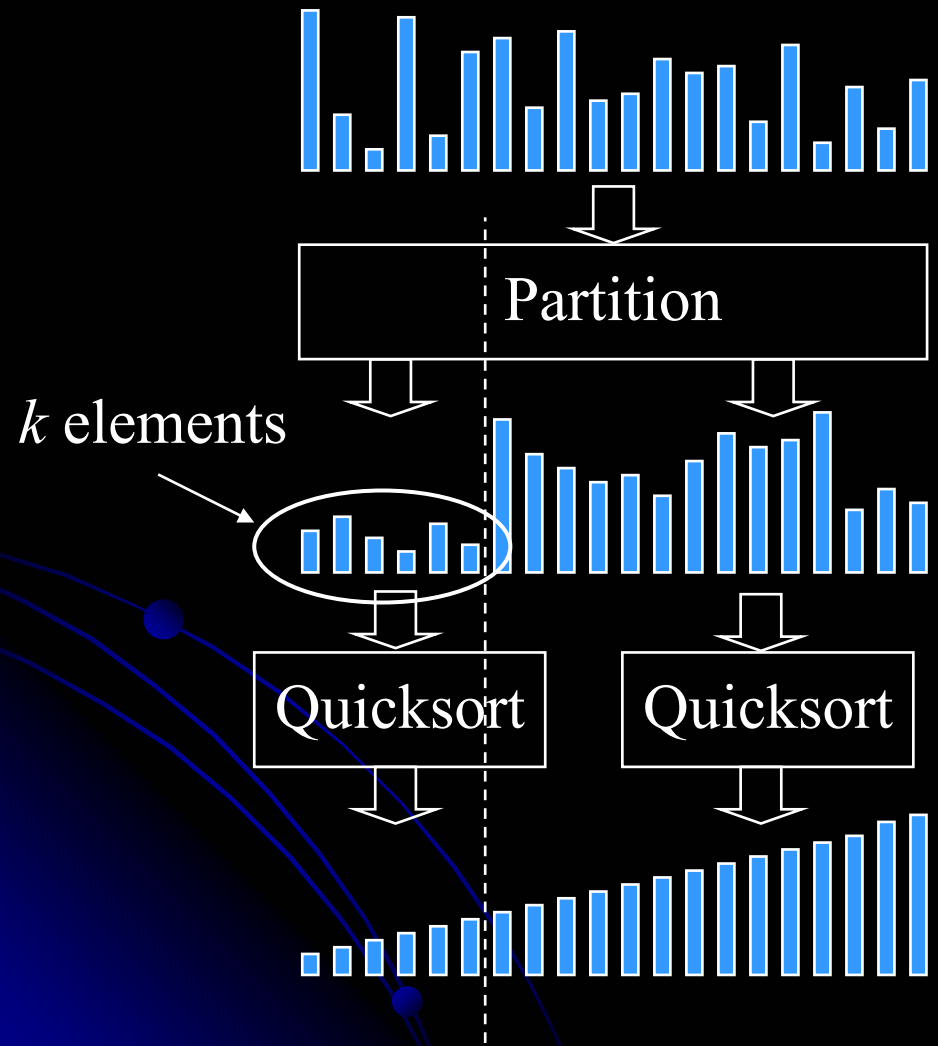
somchaip@chula.ac.th

```
void msort(int src[]) {
  int[] aux = (int[]) src.clone();
  msort(aux, src, 0, src.length - 1);
}

void msort(int src[], int dest[], int left, int right) {
  if (left < right) {
    int mid = (left + right) / 2;
    msort(dest, src, left, mid);
    msort(dest, src, mid+1, right);

    int i = left, p = left, q = mid+1;
    for (; i <= right; i++) {
      if (q > right || p <= mid && src[p] <= src[q]) {
        dest[i] = src[p++];
      } else {
        dest[i] = src[q++];
      }
    }
  }
}
```

Merge

somchaip@chula.ac.th

Quicksort



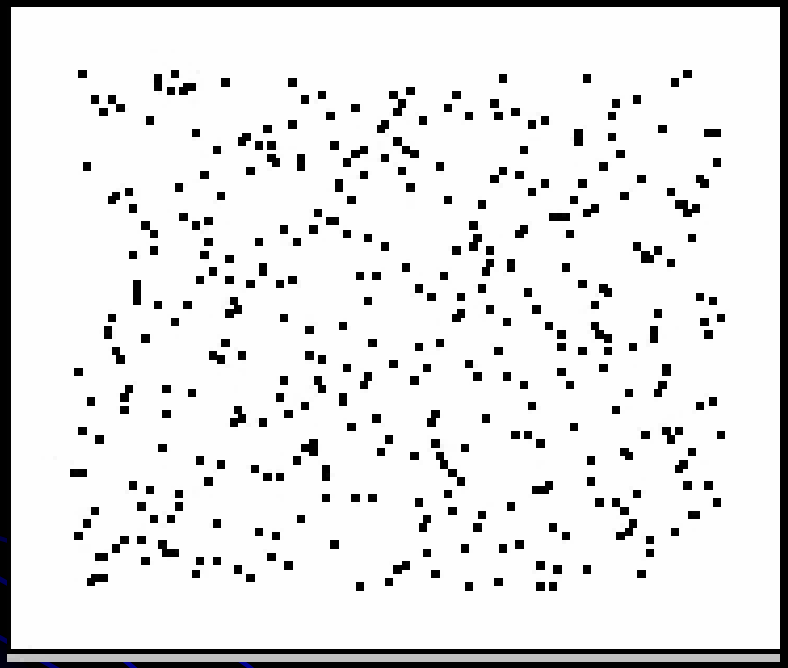
$\Theta(n)$

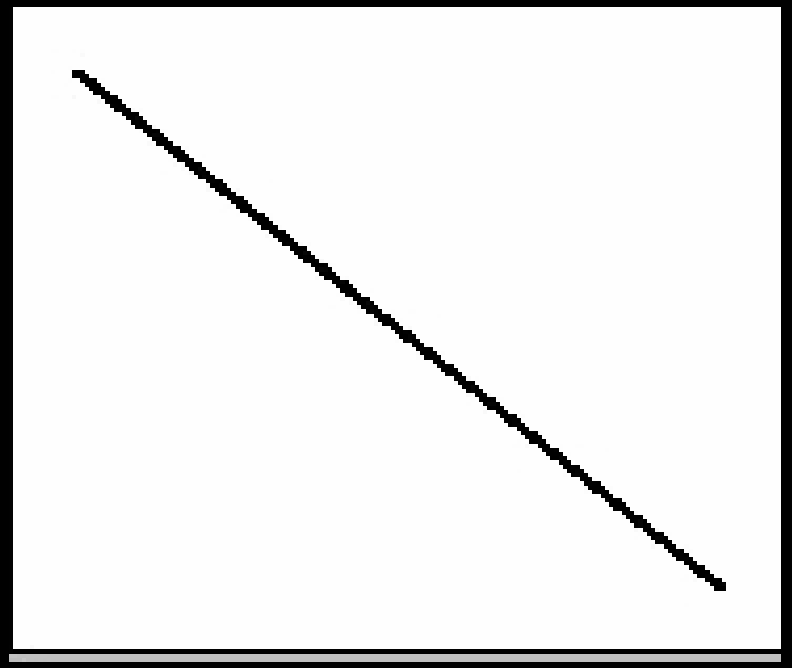
$$T(n) = T(k) + T(n-k) + \Theta(n)$$

$T(k) + T(n-k)$

somchaip@chula.ac.th

Quicksort





somchaip@chula.ac.th

Quicksort : Worst-Case Analysis

- $T(n) = T(k) + T(n\text{-}k) + \Theta(n)$
- Worst-case when $k = 1$ in every step

$$
\begin{aligned}
T(n) &= T(1) + T(n-1) + \Theta(n) \\
&= T(n-1) + \Theta(n) \\
&= \sum_{i=1}^{n} \Theta(i) \\
&= \Theta\left(\sum_{i=1}^{n} i \right) \\
&= \Theta\left(n^2\right)
\end{aligned}
$$

somchaip@chula.ac.th

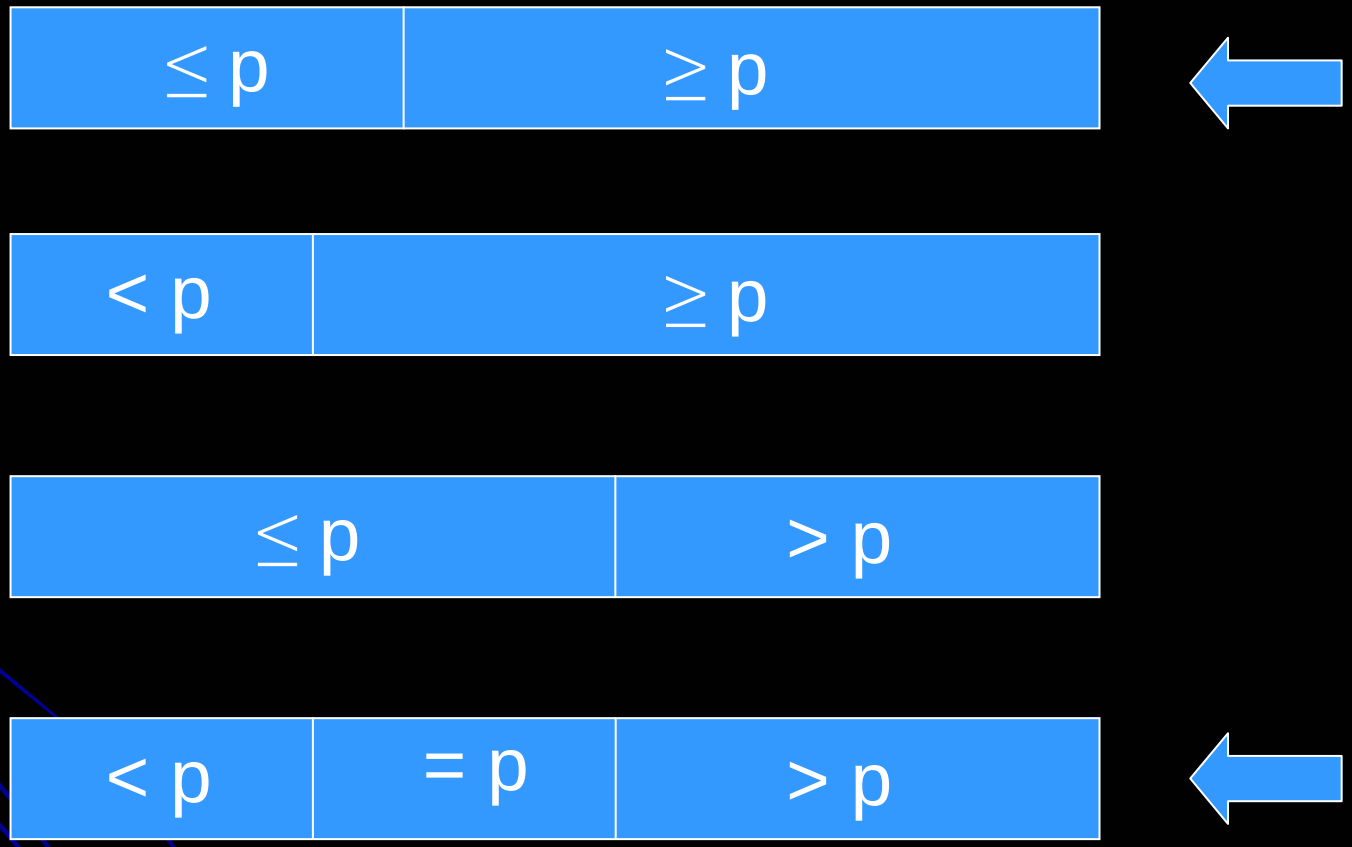
Quicksort : Best-Case Analysis

- $T(n) = T(k) + T(n\text{-}k) + \Theta(n)$
- Base-case when $k = n/2$ in every step

$$
\begin{aligned}
T(n) &= T(n/2) + T(n/2) + \Theta(n) \\
&= 2T(n/2) + \Theta(n) \\
&= \Theta(n \log n)
\end{aligned}
$$

somchaip@chula.ac.th

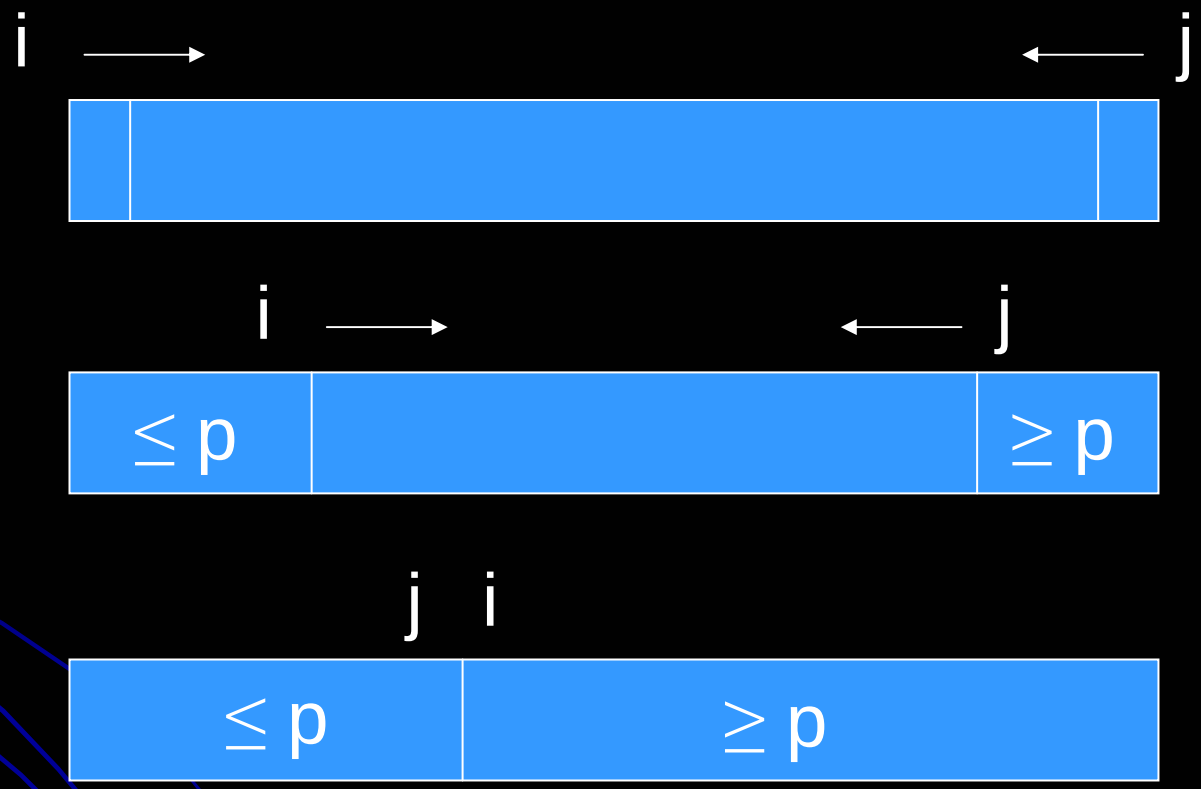
Quicksort : Partition



p - pivot

somchaip@chula.ac.th

Quicksort : Partition



somchaip@chula.ac.th

Quicksort

```
void qsort(int d[], int left, int right) {
  if (left < right) {
    int p = d[left];
    int i = left - 1, j = right + 1;
    while (i < j) {
      while (d[++i] < p);
      while (d[--j] > p);
      if (i < j) swap(d, i, j);
    }
    qsort(d, left, j);
    qsort(d, j + 1, right);
  }
}
```

Partition

somchaip@chula.ac.th

Tuned Quicksort (Bentley)

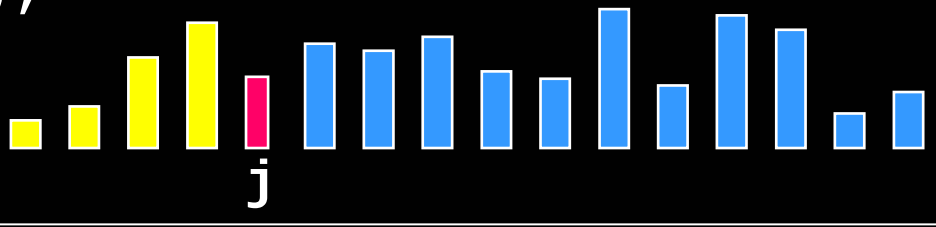
- ใช้ insertion sort เมื่อ n < 7
- ถ้า 7 ≤ n < 40
  - เลือก pivot จาก median ของตัวซ้าย กลาง ขวา
- ถ้า n > 40
  - เลือก pivot จาก median of median of three ของข้อมูล 9 ตัว
- partition แบบ 3 ช่วง <p ==p >p

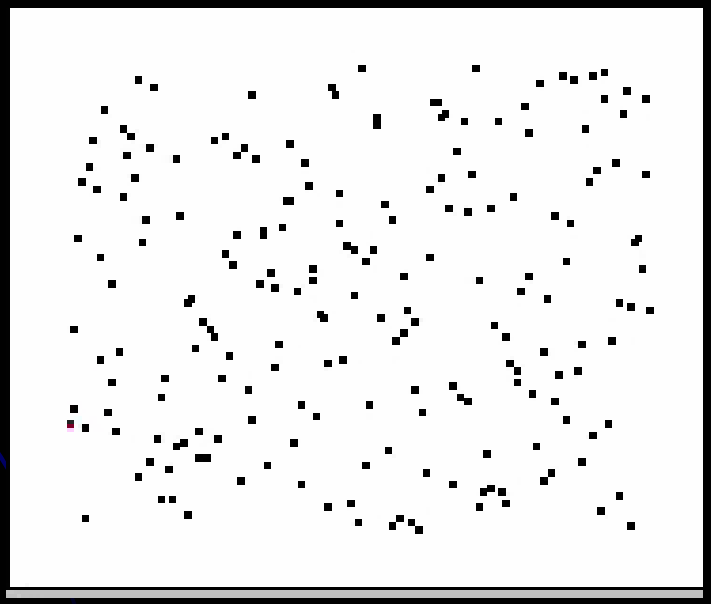
Jon L. Bentley and M. Douglas McIlroy's "Engineering a Sort Function", Software-Practice and Experience, Vol. 23(11) P. 1249-1265 (November 1993).

somchaip@chula.ac.th

Insertion Sort

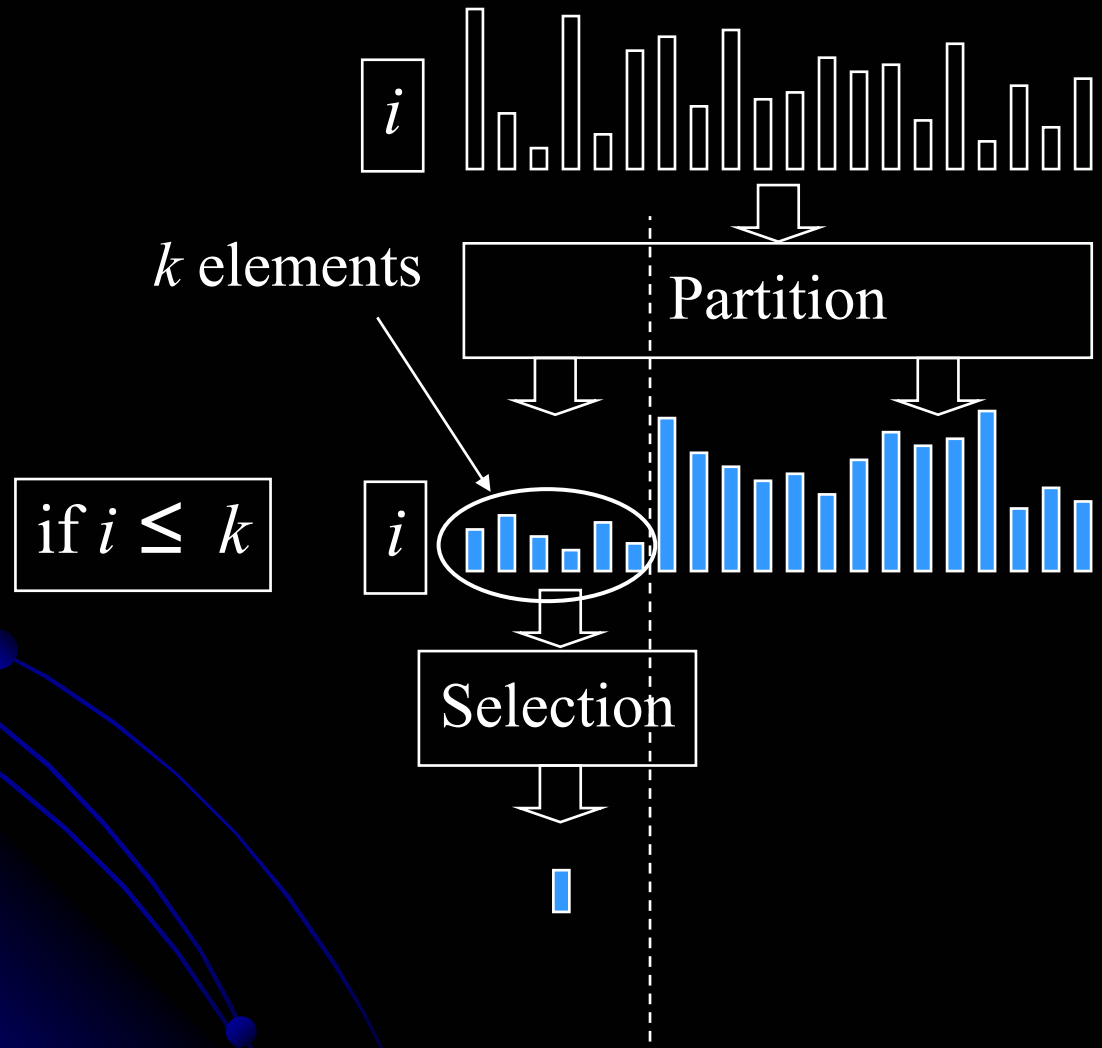
```
void isort(int d[]) {
  for (int i = 0; i < d.length; i++) {
    for (int j = i; j > 0 && d[j - 1] > d[j]; j--) {
      swap(d, j, j - 1);
    }
  }
}
```





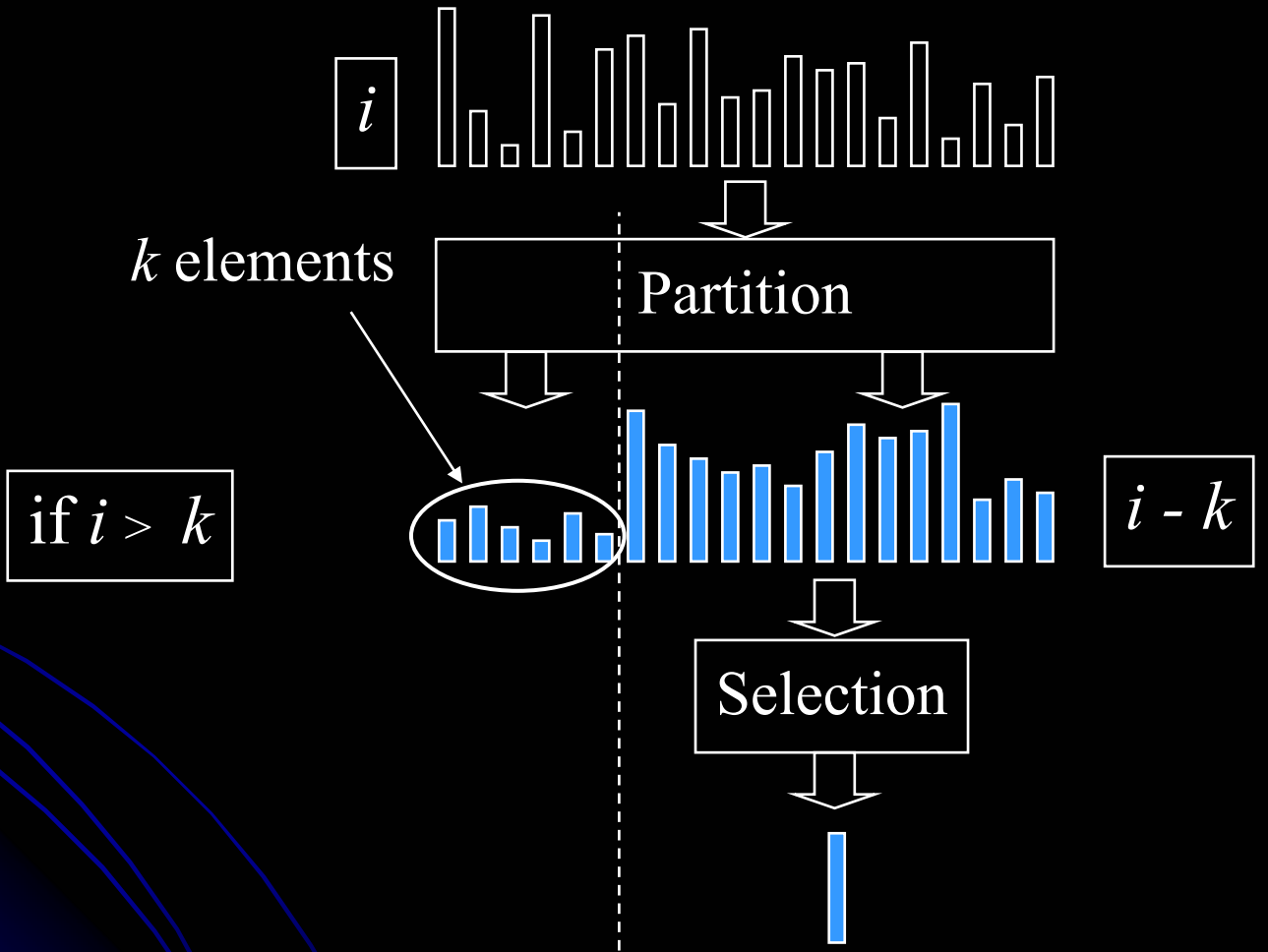
somchaip@chula.ac.th

Selection



somchaip@chula.ac.th

Selection



somchaip@chula.ac.th

Binary Search

```
int bSearch(int[] a, int key) {
  int left = 0;
  int right = a.length - 1;
  while (left <= right) {
    int mid = (left + right) >> 1;
    if (a[mid] < key) {
      left = mid + 1;
    } else if (a[mid] > key) {
      right = mid - 1;
    } else {
      return mid; // key found
    }
  }
  return -(left + 1); // key not found.
}
```


somchaip@chula.ac.th

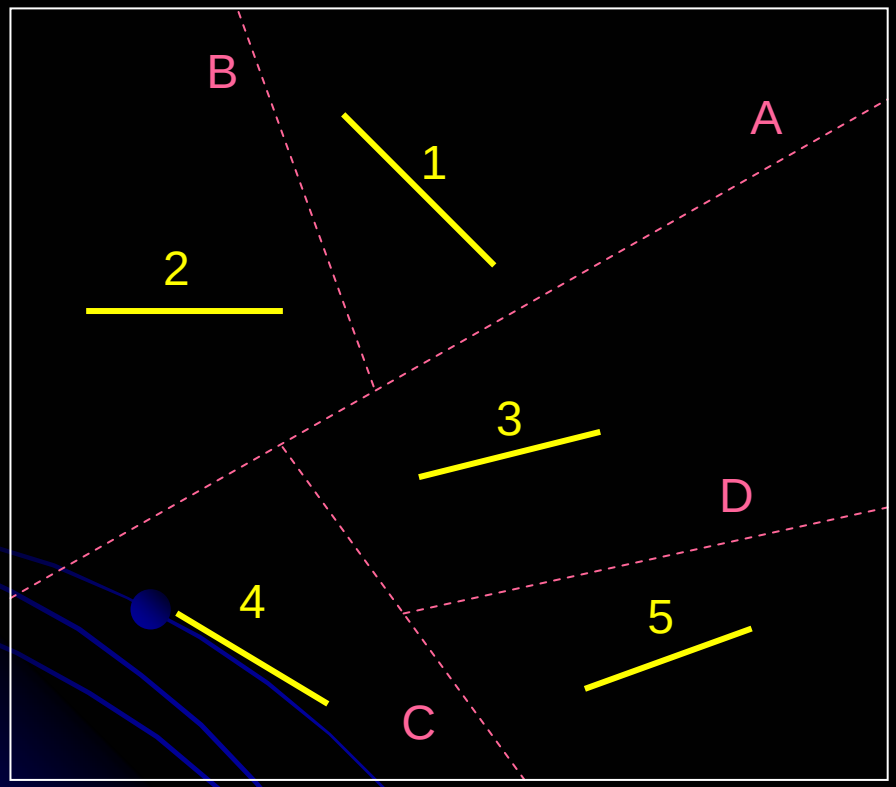
Binary Search

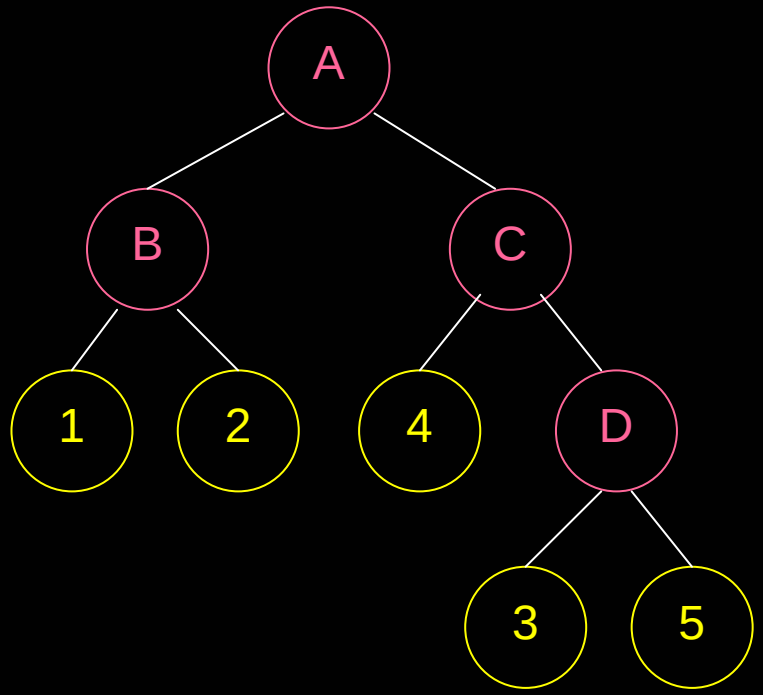
```
int bSearch(int[] a, int key) {
  return bSearch(a, key, 0, a.length - 1);
}

int bSearch(int[] a, int key, int left, int right) {
  if (left > right) return -(left + 1);
  int mid = (left + right) / 2;
  if (key == a[mid]) return mid;
  if (key < a[mid])
    return bSearch(a, key, left, mid - 1);
  else
    return bSearch(a, key, mid + 1, right);
}
```

$$T(n) \leq T(n/2) + \mathrm{O}(1) \;=\; \mathrm{O}(\log n)$$

somchaip@chula.ac.th

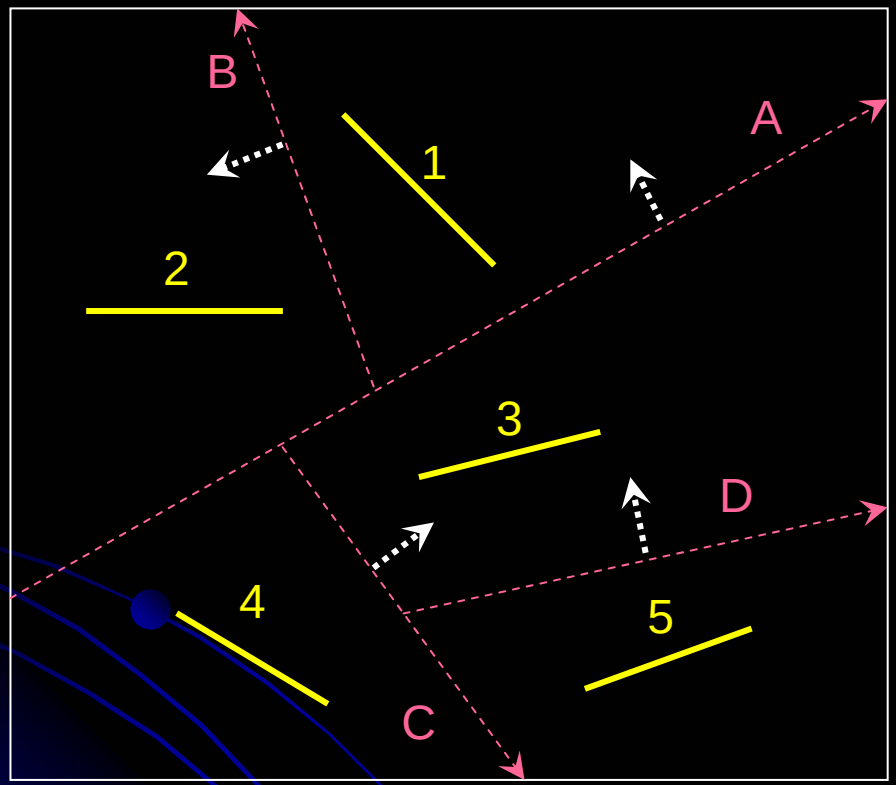
Binary Space Partition Tree

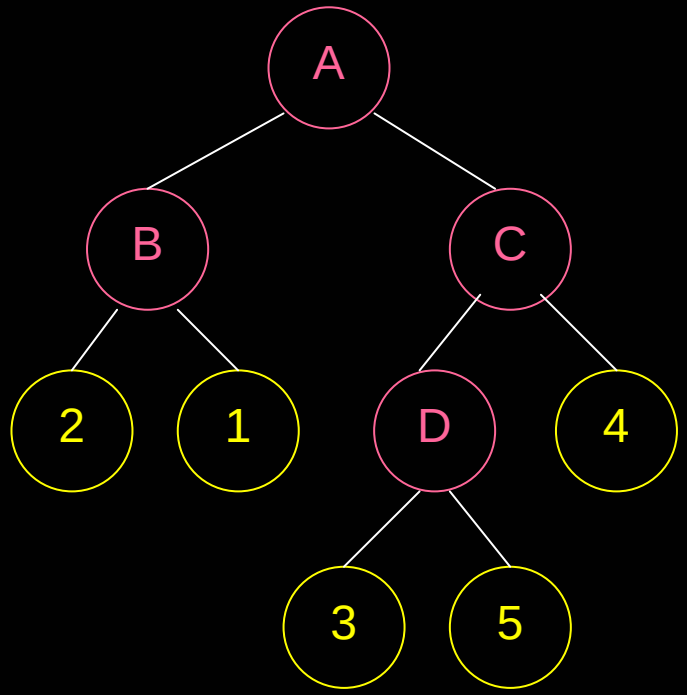




somchaip@chula.ac.th

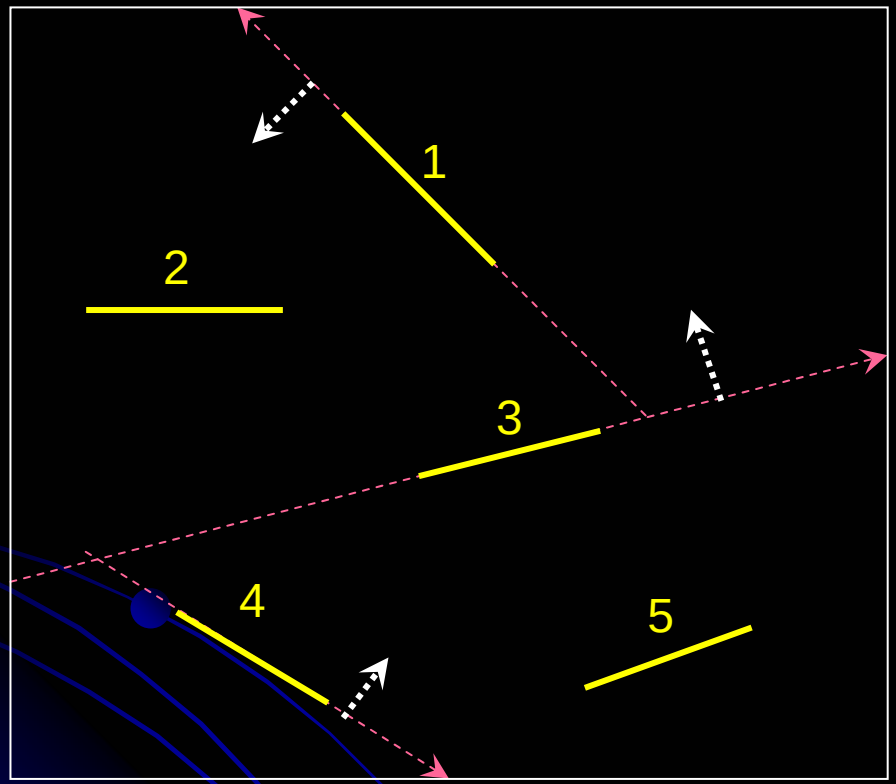
Binary Space Partition Tree

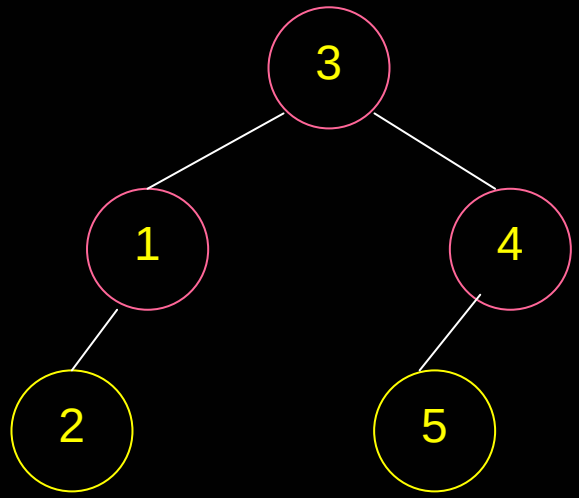




somchaip@chula.ac.th

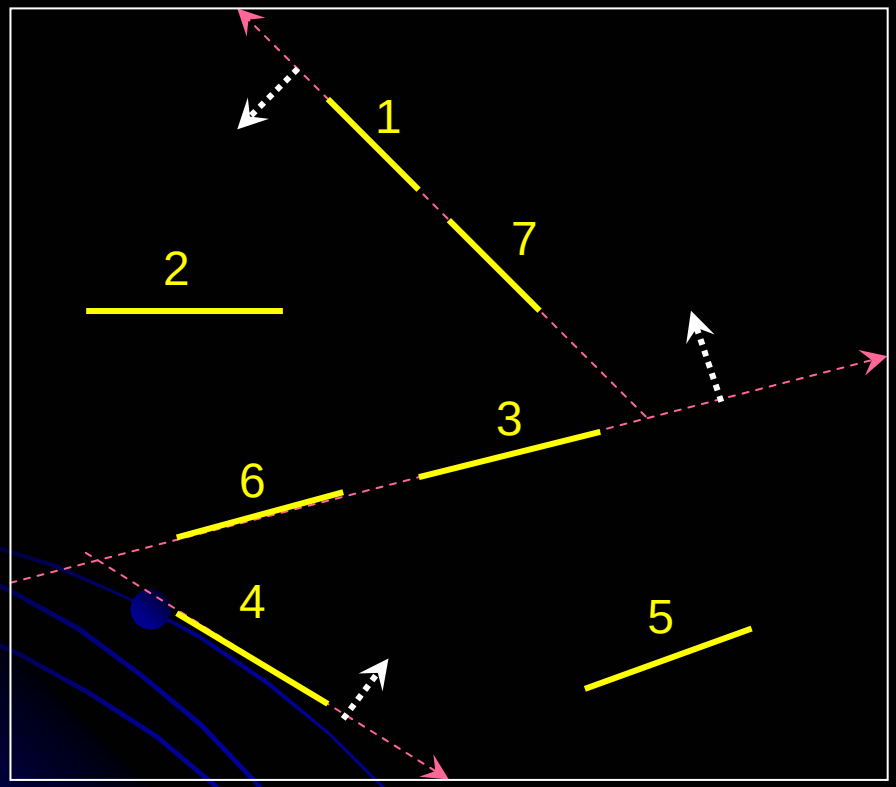
Binary Space Partition Tree

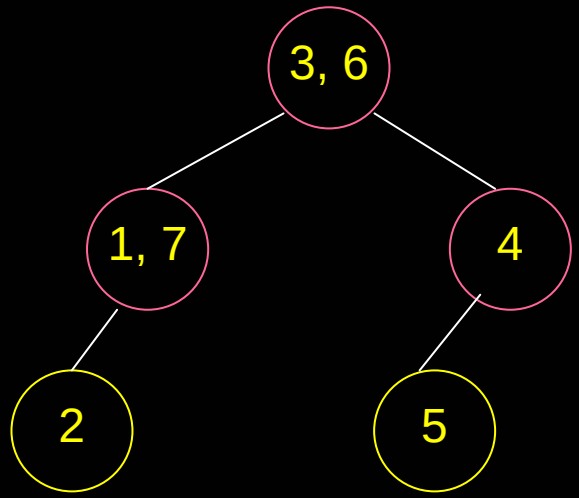




somchaip@chula.ac.th

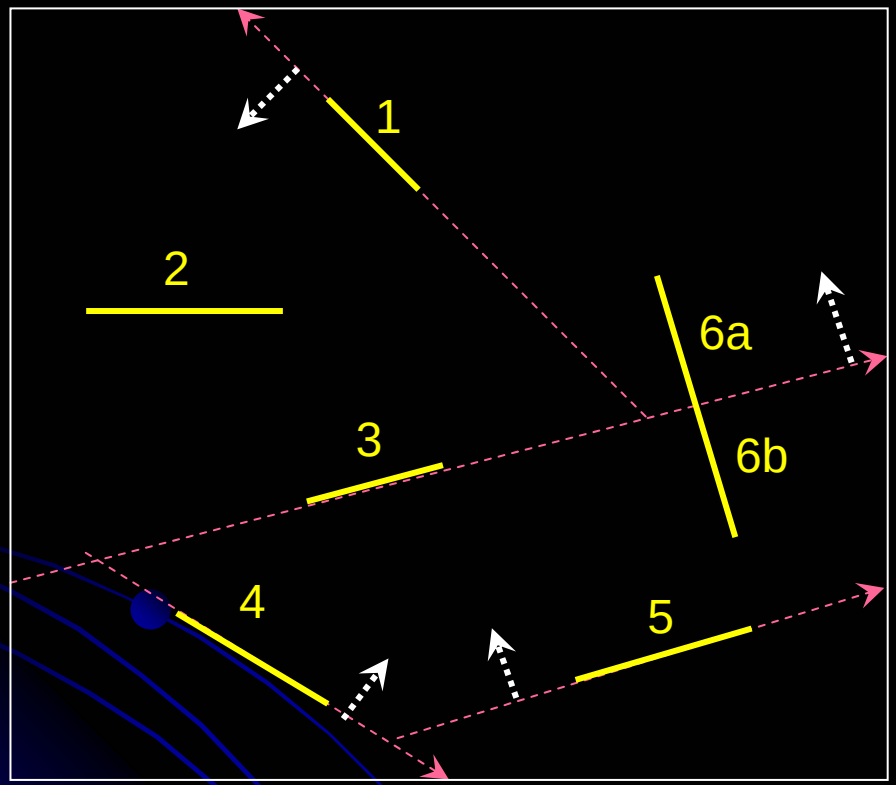
Binary Space Partition Tree

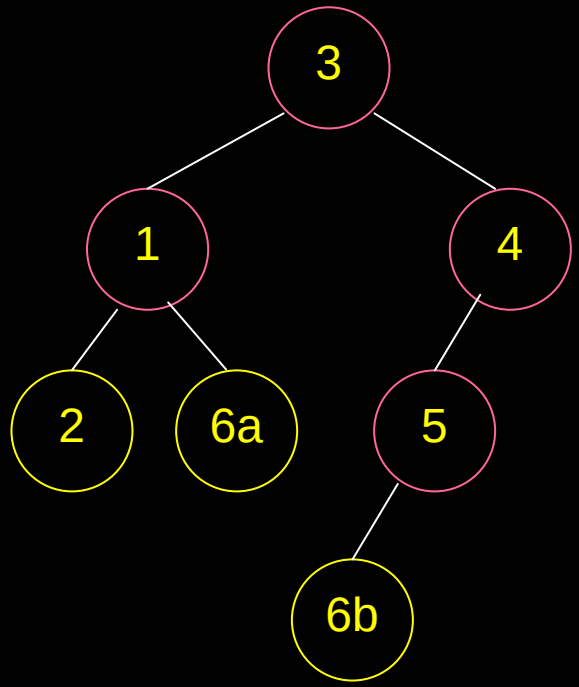




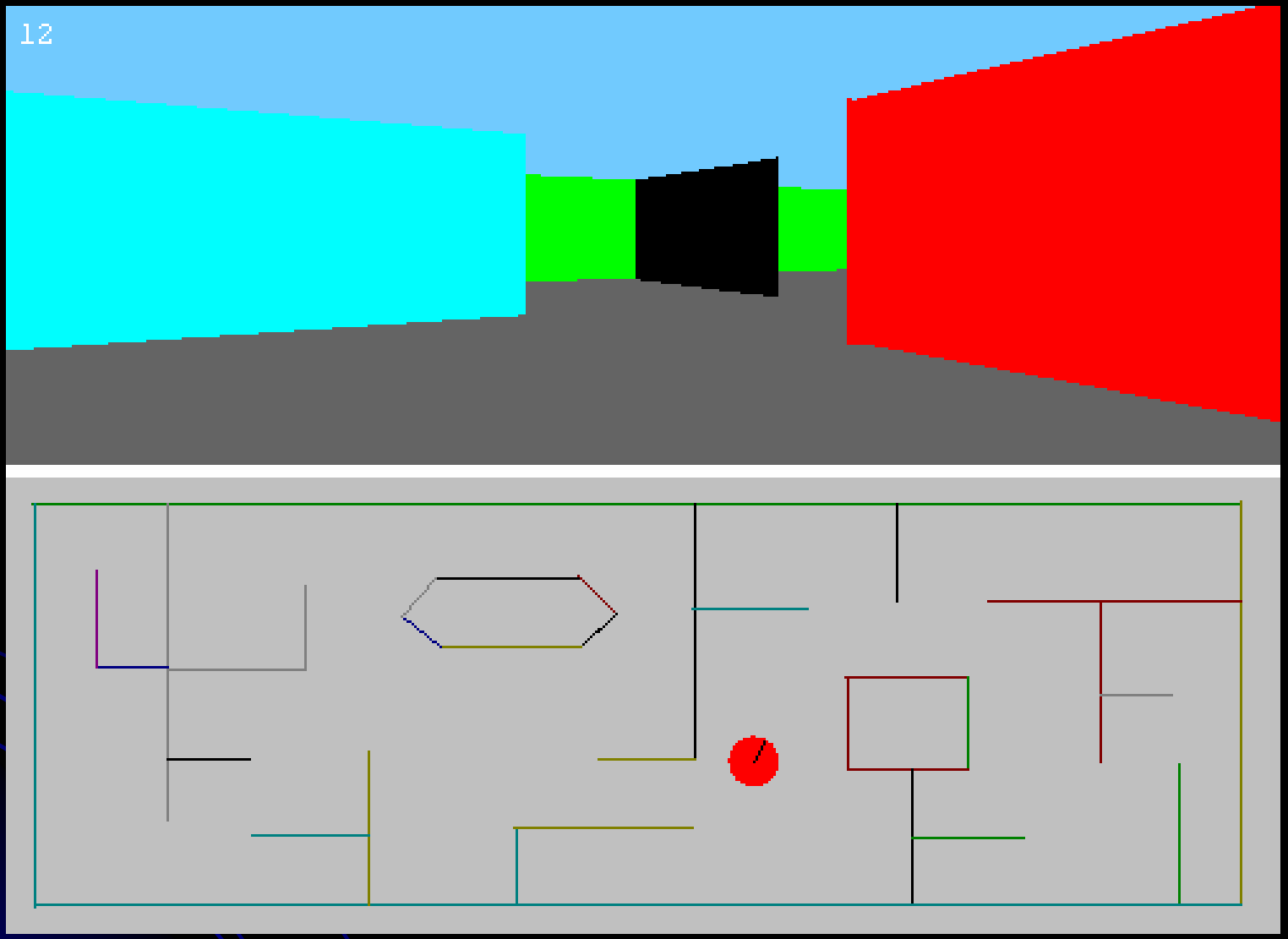
somchaip@chula.ac.th

Binary Space Partition Tree





somchaip@chula.ac.th



somchaip@chula.ac.th