Lecture 26

② Iteration method for recurrances

-decompose recurrance into a series of terms, and derive the n^{th} expression from the previous ones

Ex. Binary Search (half-interval / logarithmic search)

```
binary_search(A, lo, hi, x)
    {
        if (A[lo] > A[hi)
            return false
        mid = L(lo+hi)/2
        if (x = A[mid])
            return true
        if (x < A[mid])
            binary_search(A,lo,mid-1,x)
        if (x > A[mid])
            binary_search(A,mid+1,hi,x);
    }
```

constant time: C_1

constant time: C_2

Constant time: C_3

Same problem of size $n/2$

Same problem of size $n/2$

$A = \{1, 2, 3, 4, 5, 7, 9, 11\}$ $n = 8$

$x = 7$ $lo = 1$ $hi = 8$

1	2	3	4	5	7	9	11
1	2	3	4	5	6	7	8

$A[mid] = A[4] = 4$

$mid = \lfloor(1+8)/2\rfloor = \lfloor 4.5 \rfloor = 4$

$A[mid] = A[4] = 4$

1	2	3	4	5	7	9	11
1	2	3	4	5	6	7	8

$\text{mid} = \lfloor (5+8)/2 \rfloor = \lfloor 6.5 \rfloor = 6$

$A[\text{mid}] = A[6] = 7$

$A[\text{mid}] = A[6] = 7$

$\boxed{x = 6}$

From the binary search alg. above...

$$T(n) = C + T(n/2)$$

Find explicit formula of $T(n)$ using the iteration method

$$
\begin{aligned}
T(n) &= c + T(n/2) \quad \leftarrow 1^{ST} \text{ iteration} \\
&= c + c + T(n/4) \quad \leftarrow 2^{ND} \\
&= c + c + c + T(n/8) \quad \leftarrow 3^{RD} \\
&= \underbrace{c + c + c + c}_{4 \text{ times}} + T(n/2^4) \quad \leftarrow 4^{th} \\
&= kc + T(n/2^k)
\end{aligned}
$$

Assuming $n = 2^k$, then $k = \log n$

$\log n = \log 2^k$

$\log n = k \cancel{\log_2 2}$

$k = \log n$

$$T(n) = c \log n + T(1)$$

Thus $T(n) = O(\log n)$

Merge sort

- efficient sorting alg.
- input
  - unsorted array E
  - integers first, last
- output
  - array E containing a permutation of the input such that

$$E[first] \leq E[first+1] \leq \dots E[last-1] \leq [last]$$

Idea:

① Divide inserted list (array) into n sublists, each containing one element (a list of one element is considered sorted)

② Repeatedly merge sublists to produce new sorted sublists until there is 1 sublist remaining (this is the sorted list)

Merge sort uses the "merge" func., built into C++ STL

```
merge_sort(E, first, last)
    {
        if (first<last)
            mid=floor(first+last)/2;      ← constant time c
            merge_sort(E,first,mid);      ← same problem size n/2
            merge_sort(E,mid+1,last);     ← same problem size n/2
            merge(E,first,mid,last);      ← call merge = time n
    }
```

* Drop constant time c in favor of larges time n

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

Now use iteration to solve

$$T(n) = n + 2T(n/2)$$
$$= n + 2(n/2 + 2T(n/4))$$

$n + n + 4T(n/4)$

$$= n + n + 4\left(\frac{n}{4} + 2T\left(\frac{n}{8}\right)\right)$$
$$= n + n + n + 8T\left(\frac{n}{8}\right)$$
$$= 3n + 2^3 T(n/2^3)$$
$$= 4n + 2^4 + T(n/2^4)$$
$$= kn + 2^k T(n/2^k)$$