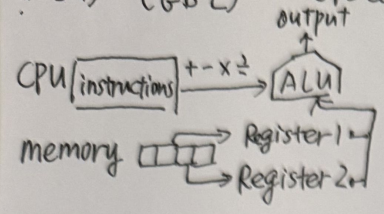
12312710 DSAA CHEATING PAPER

CPU $\xleftarrow[\text{cost}]{\text{CPU}}$ RAM $\xleftarrow[\text{cost}]{\text{I/O}}$ Harddisk

(ALU) (GBL) (TB Level)



memory: finite sequences of cells each has same bits. ascendingaddres (store infor for immediate use)

CPU: contains a fixted nums of reg.

Basic(Atomic) operations:

- Initialization — set reg to values
- Arithmetic(ALU)

Take integers from 2 reg, calculate by + - x ÷ and store result in a reg.

- Comparison. Take 2, compare
- Memory Access, Take a memory address stored in reg. Do read/write

---

Big-O notation (比快)

$\exists C_1>0, \forall n \geq C_2 \; (f(n) \leq C_1 \cdot g(n))$

C_2

存在 C_1，使得所有大于 C_2 的 n 都有 $f(n) \leq C_1 \cdot g(n)$，(一般来说，C_1, C_2 取大即可)

Big-Ω notation (比慢)

$\exists C>0, C_2, \forall n \geq C_2 (f(n) \geq C \cdot g(n))$

if $f(n)=O(g(n))$, $f(n)=\Omega(g(n))$

↓ $f(n)$ 比 $g(n)$ 慢 $f(n)$ 更快

then $f(n)=\Theta(g(n))$ 一样快

Binary search

```
left ← 1, right ← n
repeat
    mid ← (left+right)/2
    if (t = mid) return TRUE
    else if (t < mid) right ← mid-1
    else if (t > mid) left ← mid+1
until left > right
return False
```

Worst time for Binary search

$g(n) = 2+9(1+\log_2 n) = \Theta(\log n)$

Selection Sort (n^2) worse n^2, best n^2, not stable

~~选择sort有点像在已经排好的部分的基础上插入一张新的牌~~

```
for i ← 1 to n-1
    k ← i
    for j ← i+1 to n
        if A[k] > A[j]
            k ← j
    swap A[i] and A[k]
```

第一个位置：找出最小的放入

2：第二小的放入

3：…… 前 i 个是严谨正确的

Insertion Sort (n^2) worse n^2, best n

插入sort像插牌 Stable

```
for i ← 1 to n
    for j ← i to 1, >1
        if A[j-1] > A[j] swap A[j-1], A[j]
        else breaks
```

Bubble sort (n^2), W n^2, b n. Stable

```
for integer i ← 1 to n-1
    for j ← 2 to n
        if A[j-1] > A[j] swap A[j-1] A[j]
```

$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) \leq O(n^2) < O(2^n) < O(n!) < O(n^n)$

Mergesort(A[], n) always nlogn stable

```
if n>1  p ← ⌊n/2⌋
B[1...p] ← A[1...p]
C[1...n-p] ← A[p+1, n]
Mergesort(B, p)
Mergesort(C, n-p)
A[1...n] ← Merge(B, p, C, n-p)
```

```
Merge(L, nL, R, nR)
n ← nL+nR
A ← new int[n]
i ← 1; j ← 1
for k ← 1 to n
if (i ≤ nL) & (j > nR | L[i] < R[j])
    A[k] ← L[i]; i++
else A[k] ← R[j]; j++
return A
```

```
Quicksort(A[1...n], lo=1, hi=n)
    p ← partition(A, lo, hi)
    Quicksort(A, lo, p-1)
    Quicksort(A, p+1, hi)
partition(A, lo, hi)
    p ← random(lo, hi)
    pivot ← A[p]
    L ← lo, R ← hi
    for i lo to hi
     if (i != p)
       if (A[i] < pivot) A'[L++] ← A[i]
       else A'[R--] ← R[i]
    A'[L] ← pivot
    A[lo, hi] ← A'
    return L;
```

arr: nlogn worse: n^2 best: n not stable

Master Theorem

$T(n) = aT(\frac{n}{b}) + O(n^\gamma)$

if $\log_b a < \gamma$ then $T(n) = O(n^\gamma)$

if $\log_b a = \gamma$ $T(n) = O(n^\gamma \log n)$

if $\log_b a > \gamma$ $T(n) = O(n^{\log_b a})$

Binary $T(n) \leq T(\frac{n}{2}) + C_2$

Merge $T(n) = 2T(\frac{n}{2}) + O(n)$

数组：① O(1)时间访问任意项 ② 可以随机读写 ③ 内存紧凑

① 必须初始大小 ② 很难调整大小 ③ 很难插入/删除

链表 ① 无限制生长 ② 易插入/删除

① 无法随机访问 ② 要.next操作 ③ 占用额外内存 4+4 bytes ④ 不紧凑

遍历链表

```
traverse(A)
  if(A==null) return
  else print A.value
    traverse(A.next)
```

要倒叙输出则对接这面

找、删、更新指定i，都用O(n)

Stack → 先进后出

Queue → 先进先出

Brute Force 暴力

O(mn)

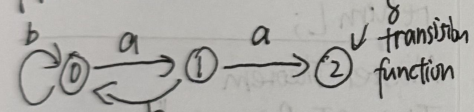
Rabin-Karp

把字串map到一个数值上

再对数值比较

正常O(n) worse O(mn)

Finite State Automata



Q={0,1,2}, q_0=0, 从q_0开始

Σ={a,b}.

abaabaca 的next[];

next[0] null "a" 0

next[1] "b" 0

next[2] "aba" 1

next[3] "abaa" 1

next[4] "abaab" 2

next[5] "abaaba" 3

next[6] "abaabac" 0

next[7] "abaabaca" 1

↓

最长前后公共串

生成next：

```
next[] = int[length]
le ← 0
ri ← 1
while ri < length
  if(string.(lef) = String(rig){
    next[rig] = lef+1
    rig.lef++
  }else if(lef==0)
    next[rig]=0, rig++
  else lef=next[lef-1]
```

Transition function

转移方程（表格）

	0	1	2	3	4	5	6	7
a	1		3	4			6	8
b	0	2				5		
c	0						7	

第一步，先把位置填上

	0	1
a	1	?
b	0	2
c	0	0

第二步：第0个function

对应的是next[0]

next[0]=0，复制第0行

	0	1	2	3	4	5	6	7
a	1	1	3	4	1	6	4	8
b	0	2	0	2	5	0	2	0
c	0	0	0	0	0	6	7	0

第6个function对应next[5]

next[5]=3，复制第三行