算法分析

1. 评估算法效率. 用 N 表示问题的规模(输入数据)

运行时间 $f(N)$. N很大时称作计算复杂度.

2. 选择排序. 2个循环. $1+2+3+\cdots+n=\frac{n^2+n}{2}$

只考虑: 执行多少轮循环. 不考虑循环内指令效率.

3. 大O表示性能 ← $O(f(n))$. 去掉常数因子.

选择排序 $O(\frac{n^2+n}{2})$ ✗ n很大时. $n^2 \gg n$. → $O(\frac{n^2}{2})$

$n^2 \simeq \frac{n^2}{2}$ → $O(n^2)$

$O(N^2)$ 特征值而不是实际值.

↘ quadratic time.

数据规模 x2 → running time x4.

4. $O(\log_2 n)$ 二分搜索

$O(n)$ 线性搜索

$O(n\cdot\log_2 n)$ 归并. 快排

$O(n^2)$ 选择. 冒泡.

不存在线性排序

插入排序?

5.

```
function insertionSortR(array A, int n)
    if n>0
        insertionSortR(A,n-1)
        x ← A[n]
        j ← n-1
        while j >= 0 and A[j] > x
            A[j+1] ← A[j]
            j ← j-1
        end while
        A[j+1] ← x
    end if
end function
```

伪代码 → 论文

递归次数 n.

while 循环: worse case: n

∴ $O(N^2)$

归并: 分: $\log_2 n$ 合: n → $O(n\cdot\log_2 n)$

N	Selection sort	Merge sort
10	0.00013	.00094
100	0.00967	.012
1000	1.08	.14
10,000	110.0	1.6

6. Test time

1970-01-01 00:00:00
↓ $+2^{32}$s
2038年. OS开始

```
#Include <time.h>
time_t then= time(0)
printf("%llu\n", time(0)-then);
```

```
4  int main()
5  {
6      time_t then = time(0);
7      printf("%lu\n", then);
8      for ( int i=0; i>=0; i++ ) {
9          // for ( int j=0; j>=0; j++ ) {
10         // }
11     }
12     time_t now = time(0);
13     printf("%lu %lu\n", now, now-then);
14 }
```

wall clock time

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
ZJUCS:cc % gcc t.c
ZJUCS:cc % ./a.out
1590979477
1590979483 6
```

```
ZJUCS:cc % time ./a.out
1590979552
1590979558 6
./a.out  4.54s user 0.18s system 77% cpu 6.056 total
```

占用77% cpu

用time测运行时间

程序工作　操作系统

无法得到准确时间.

还要把别的程序.

结构化程序设计 ← (小结构) 代码段

≠一连串语句

函数=若干个结构. 粒度大.

1. 早期程序

- 早期的计算机存储器容量非常小，人们设计程序时首先考虑的问题是如何减少存储器开销，硬件的限制不容许人们考虑如何组织数据与逻辑，程序本身短小，逻辑简单，也无需人们考虑程序设计方法问题。

一连串指令→程序.

一直执行.

跳转是唯一控制结构.

2 结构化程序设计

顺序. 选择. 循环.

入口. 出口单一.

控制结构为单位.

句子{表达式. 语句

```
int a,b;
int min;

scanf("%d %d", &a, &b);
if ( a<b ) {
    min = a;
} else {
    min = b;
}
int ret = 0;
int i;
for ( i = 1; i < min; i++ ) {
    if ( a%i == 0 ) {
        if ( b%i == 0 ) {
            ret = i;
        }
    }
}
printf("%d和%d的最大公约数是%d.\n", a, b, ret);
```

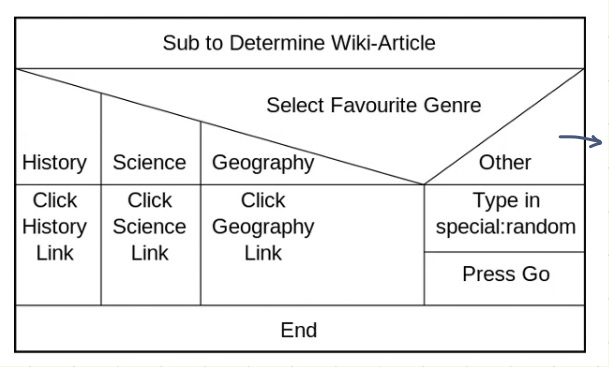
表达式　语句　语句　语句

(可以赋值)

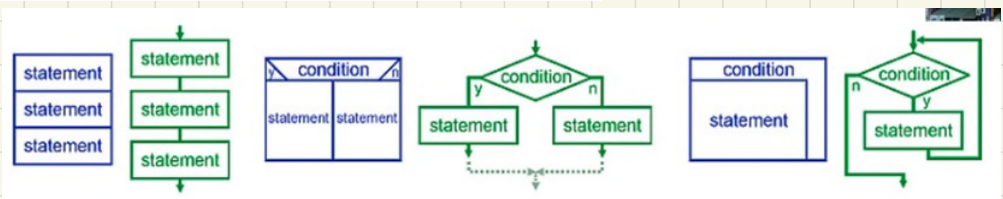
i = xx ✓ → xx是表达式

3. 流程图表达不合理.

Nassi-Shneiderman图



→ switch



4. 面向对象 (略)

5. 变量
- Global
- Local
- Static
- Allocated

```
int i;              // global vars.
static int j;       //static global vars.
f() {
    int k;          // local vars.
    static l;       // static local vars.
    int *p = malloc(sizeof(int));
  //allocated vars.
}
```

p是Local　　*p是Allocated

变量储存