数据结构

- 【第一道选择题-时间复杂度】：①嵌套循环应全部标注出每个嵌套循环的总循环次数；②取$n = 2^{100}$算出循环次数。
  - ◇ 关于树的遍历一般都会访问所有结点，时间复杂度为$O(n)$。
- 【算法的特性】：有穷性、确定性、可行性、零个或多个输入、一个或多个输出。
- 可以用一个队列模拟栈，但**不能用一个栈模拟队列**。
- 【两个队列模拟栈】：插入时将元素插入到空队列中，然后将另一个队列的所有元素依次出队并入队到该空队列。
  - ◇ 入栈：①新元素入队到空队列 q2；②将 q1 中的所有元素依次出队并入队到 q2；③交换 q1 和 q2；
  - ◇ 出栈：直接从 q1 出队。
- 【两个栈模拟队列】：栈 s1 用于插入元素，栈 s2 用于删除元素。
  - ◇ 入队：直接将新元素压入栈 s1；
  - ◇ 出队：①若栈 s2 不为空，则直接从 s2 弹出元素；②若栈 s2 为空，则将 s1 中的所有元素弹出并压入到 s2，然后从 s2 弹出元素。
- 【前序遍历和中序遍历的前驱后继】：前驱为**左子树中最右的结点**或祖先结点中离它最近的**左上方祖先**；后继为**右子树中最左的结点**或祖先结点中离它最近的**右上方祖先**。
- 【后序遍历的前驱后继】：前驱是**父节点的左子树的最右结点**，后继是**父节点的右子树的最左结点**。
- 【前序序列和后序序列确定**祖先关系**】：XY分别为任意长度结点组合，若在前序遍历中为$\ldots XY\ldots$，后序遍历中为$\ldots YX\ldots$时，X为Y的祖先。
  - ◇ 【前序序列和后序序列相同】：仅空树或者只有根节点的二叉树，**非空时唯一**；
  - ◇ 【前序序列和后序序列相反】：二叉树的高度和其节点个数相同，共2^{n-1}种形态，n为节点个数。
- 【二叉哈夫曼树】：①总结点数始终为$2n-1$，其中n为叶节点数，没有度为一的结点；②哈夫曼树不唯一，但是WPL必然最优。
  - ◇ 看清楚是不是二叉哈夫曼树，若为【度为m的哈夫曼树】就是m叉哈夫曼树。
- 【最小生成树】：**边数比顶点数少一**、减去一条则不连通、 增加一条则会出现回路。
- 只要出现了折半查找的相关字眼，就要联想到折半查找判定树。

● 【红黑树】：①**红黑树是能自平衡的二叉排序树不是平衡二叉树**；②根结点是黑色、不存在两个相邻的红结点、从**任一结点**到任一叶结点的简单路径上的黑高相等、查找插入删除的时间复杂度都是$O(\log_2 n)$；③从根到叶结点的最长路径不大于最短路径的两倍；④内部结点数最少时为**全黑满树形态**，结点数至少为2^h-1。

● 【红黑树插入】：其中若没有叔叔结点则默认为黑色。

1. 若插入结点为根结点则染为黑色、否则染为红色，若**其父结点是红色的则需要调整**；
2. 叔叔结点为红色，则**直接染色：叔叔结点、父亲结点和祖父结点三个结点**染色，**祖父结点视为新结点**；
3. 叔叔结点为黑色，则**旋转后染色**：LL型右单旋、RR型左单旋、LR型左右双旋、RL型右左双旋，旋转后将**父亲结点和祖父结点两个结点**染色。

▲ 因为此时可能没有叔叔结点所以不用染色。

● 【折半查找判定树】：①**折半查找相关的题都应该转化到折半查找判定树上**；②折半查找的平均查找长度即为折半查找判定树的ASL=($\sum_{i=1}^{n}$第i层的成功结点数$\times i$)÷成功结点总数；③查找成功的最大路径长度为$\lceil\log_2(n)\rceil$，查找失败最大路径长度为$\lceil\log_2(n+1)\rceil$；④折半查找判定树是二叉查找树（中序序列是有序序列）、不一定是完全二叉树但一定是一个**平衡二叉树**；⑤**任意一结点右子树结点数最多比左子树结点数多一个**（|右−左|= 0或1）。

● 【二叉树的顺序存储】：**存储一棵高度为h且有m个结点至少需要2^h-1个存储单元**。

● 【二叉树的链式存储】：在含有n个结点的二叉链表中，含有$n+1$个空链域。

● 排序算法总结

◇ 每趟排序结束都**至少能够确定一个元素最终位置**：简单选择排序、快速排序、堆排序。

◇ **最后一趟排序开始时所有元素可能都不在最终位置上**：直接插入排序。

◇ **排序趟数**与关键字的初始排列初始状态无关：插入排序$n-1$趟、选择排序$n-1$趟、基数排序d趟。

◇ **比较次数**与关键字的初始序列初始状态无关：基数排序（不需要进行关键字的比较）、选择排序（简单选择排序、堆排序，比较次数始终为$\frac{n(n-1)}{2}$）。

◇ **移动次数**与关键字的初始排列初始状态无关：**基数排序**。（**反正看到无关选基数排序一定没错**）

◇ **时间复杂度**与关键字的初始排列初始状态无关：直接选择排序$O(n^2)$、堆排序$O(n\log_2 n)$、归并排序$O(n\log_2 n)$、**基数排序**$O\big(d(n+r)\big)$。

◇ 直接插入排序和冒泡排序在最好情况（有序）下的时间开销为线性时间$O(n)$，为所有排序算法中最快的。

◇ 不稳定：堆排序、快速排序、希尔排序、直接选择排序。

● 【归并排序】：①两长度为a、b的有序表进行归并，**比较次数**最大值为$a+b-1$；②归并排序是稳定的；

● 【简单选择排序】：①从待排序序列中选择最小元素并与未排序部分的第一个元素交换；②**不稳定**，可用于链表；③比较次数始终为$\frac{n(n-1)}{2}$，与序列状态无关。

● 【快速排序】：①不稳定、会退化、有$O(n\cdot\log_2 n)$空间复杂度、不适用于小数据和有序数据；②任意一个枢纽划分后的两边**块间有序，块内不一定有序**；③快排的空间复杂度来源于其递归调用的深度；④可使用三数取中法、随机选择法或恒取首位将空间复杂度优化为$O(\log_2 n)$；⑤不产生有序子序列；⑥递归次数与**初始序列**和**选择的枢轴元素**有关，与分区处理顺序无关；

◇ 若划分的范围内只有一个元素，则不用继续比较

● 【堆排序】：①比较总次数不超过$4n$；②适合大量**数据**进行排序（但不能用于外存），不适合大**文件**排序，文件排序只能使用归并排序；

● 【堆的应用-优先队列】

◇ 数据结构定义

```
struct MaxHeap {
    int* data;          // 存储堆的数组
    int size;           // 当前堆中元素的个数};
```

◇ 插入结点

```
void insert(MaxHeap* heap, int value) {
    if (heap->size >= MAX_SIZE) {return ;}      // 堆已满
    heap->data[heap->size] = value;heap->size++;  // 将新元素插入到堆的末尾
    heapifyUp(heap, heap->size - 1);            // 向上调整堆结构}
```

◇ 移除结点

```
int removeMax(MaxHeap* heap) {
    if (heap->size == 0) return -1;       // 堆为空,返回错误值
    int max = heap->data[0];                    // 获取最大元素即优先度最高的元
素
    heap->data[0] = heap->data[heap->size - 1];  // 将堆底元素移动到堆顶
    heap->size--;heapifyDown(heap, 0);  // 向下调整堆结构
    return max;
}
```

● 【平衡二叉树的删除】

◇ 【删除结点】：①删除叶子结点，直接删除；②删除的结点只有左/右子树：用删除结点的子树顶替位置；③删除的结点既有左子树又有右子树：用删除结点的前驱/后继结点顶替，并转换为前驱/后继结点的删除；

◇ 删除后从删除的结点开始向上寻找最小不平衡子树并调整。

● 【平衡二叉树删除后又加入】：①若其为叶节点，则前后相同；②若其并非叶节点，则前后不同。

● 【※※※描述Dijkstra算法】

◇ 从源点开始分别初始化三个数组，即完成标记final，最短路径长度dist，最短路径上的前驱path，然后将与源点直连点对应的dist初始化为其长度、path初始化为零；

◇ $while$（遍历所有结点）：

▲ 所有未完成且最短路径长度值最小的结点并将其完成final = true；

▲ 更新该点相连的最短路径dist和前驱path；

● 【※※※Dijkstra算法】：①Dijkstra要会写，D ijk str a，又有ijk又有str；②使用邻接矩阵和邻接表表示时时间复杂度均为$O(|V|^2)$；③不适用于含有负权值的带权图。

● 【单源最短路径】：BFS算法（仅限于无权图）、Dijkstra算法（适用于无负权的带权图和无权图）。

● 【一般情况下，邻接表的时间复杂度小于邻接矩阵】：广度优先遍历BFS，深度优先遍历DFS，Prim算法。

● 【普里姆算法$Prim$】：①需要遍历所有**顶点**，**适用于边稠密图**；②时间复杂度为$O(|V|^2)$，边权不能为负。

● 【克鲁斯卡尔算法$Kruskal$】：①每次选择最小边只需要$O(\log|E|)$的时间，时间复杂度为$O(|E|\log_2|E|)$，故**适用于边稀疏顶点多的图**，边权不能为负；②将带权边从小到大排列然后**使用并查集**检查是否会成环，重复操作$n-1$次。

● 【弗洛伊德算法$Floyd$】：①可以用于任意两点间最短路径，边权可为负但不能有负环；②A^k中第k行和第k列的路径长度均不变且对角线元素始终为零，其表示其他结点若采用结点k作为中转是否会缩短路径。

● 【广度优先遍历BFS】：①需要**辅助队列**记录顶点信息，空间复杂度为$O(|V|)$；②可用于非带权图的单源最短路径问题，可用于获取连通分量数（等于调用次数）；③类似于二叉树的层序遍历；④**广度优先生成树的高度小于等于深度优先生成树的高度**；

- 【深度优先遍历DFS】：①需要一个**栈**用于递归，空间复杂度为$O(|V|)$；②可用于判断有向图中是否存在回路、可用于获取逆拓扑有序序列、可用于获取连通分量（等于调用次数）；③类似二叉树的先序遍历；④深度优先序列在邻接表实现时唯一，邻接矩阵实现时不唯一；⑤当且仅当某结点的所有后续结点全部出栈后该结点才会出栈。
- 【邻接表】：①无向图空间复杂度为$O(|V|+2|E|)$、有向图空间复杂度为$O(|V|+|E|)$；②适用于**稀疏图**；③邻接表的表示方式不唯一。

◇
```
typedef struct EdgeNode {          // 边表结点
 int adjvex;                       // 邻接顶点的编号
 EdgeNode* next;                // 指向下一个相邻顶点
}EdgeNode;

typedef struct VertexNode   {   // 边表结点
 int data;                         // 权值
 VertexNode * next;             // 指向下一个相邻顶点
}VertexNode;

typedef struct Graph {
 VertexNode* adjList;  // 存储所有顶点的数组
 int numVertices;      // 图中的顶点数量
 int numEdges;         // 图中的边数量
}Graph;
```

- 【十字链表】：①在稀疏矩阵和图的存储中均可使用，看清楚是哪里；②只能用于存储有向图（**只存储边**）、表示不唯一、空间复杂度为$O(|V|+|E|)$。
- 【邻接多重表】：只能用于存储无向图、表示不唯一、空间复杂度为$O(|V|+|E|)$。
- 【并查集】：存储值为负表示这个点有子结点，为非负（包括 0）表示其父结点的索引值。

◇ 合并：将一个集合的根节点的父节点指向另一个集合的根节点。时间复杂度为$O(1)$。

◇ 查找优化-路径压缩：当递归地查找到根节点时，依次将路径上的每个节点的父节点更新为根节点。**只压缩一条路径**。通过路径压缩优化，可以将**查找**操作的时间复杂度接近于常数级别，特别适用于大规模的并查集操作。

◇ 合并优化-按秩合并：经过路径压缩后，根节点存储值的绝对值表示这个集合的大小（包括根节点本身），可通过其考虑两个集合的大小，将节点数少的根节点合并到节点数多的根节点上，并更新节点数。

● 【并查集应用】

1. 判断图的连通分量数：遍历各边，有边相连的两个顶点确认连通，“并”为同一个集合。只要是相互连通的顶点都会被合并到同一个子集合中，相互不连通的顶点一定在不同的子集合中。
2. $Kruskal$算法的最小生成树—各边按权值递增排序，依次处理：判断是否加入一条边之前，先查找这条边关联的两个顶点是否属于同一个集合（即判断加入这条边之后是否形成回郡），若形成回路，则继续判断下一条边；若不形成回路，则将该边和边对应的顶点加入最小生成树T，并继续判断下一条边，直到所有顶点都已加入最小生成树T。

● **阶为 3 和 4 的 B 树单个非根结点最少有一个关键字，阶为 5 的 B 树单个非根结点最少有两个关键字**。

● B 树查找

◇ 在B树中找结点：在磁盘上进行。在找到目标结点后，先将结点信息读入内存。

◇ 在结点内找关键字：在内存中进行。在结点内采用顺序查找法或**折半查找法**。

● B 树插入：若导致原结点关键字数量超过上限溢出（$m-1$个关键字），从中间位置$\lceil\frac{m}{2}\rceil$（如果m为偶数则默认是$\lceil\frac{m}{2}\rceil-1$）分开，插入位置一定是最底层的某个非叶结点。**插入分裂结束后左部分的结点数少于等于右部分**。

◇ 最坏情况下，插入 B 树共需$3h+1$次操作。从根节点查找到底需要h次，每次插入后分裂需要进行两次操作并逐层向上传导，**此时会导致 B 树高度加一**。

● B 树删除

◇ 若被删除关键字在终端结点，且结点关键字个数不低于下限，则直接删除该关键字，并移动后面的关键字；

◇ 若被删除关键字在非终端结点，则用直接前驱或直接后继来替代被删除关键字，然后后面的元素直接前移；

◇ 若被删除关键字在终端结点，但是结点关键字个数删除后低于下限，**左右兄弟够借（补位）**

1. 将原结点在父结点对应连接的前/后一个关键字下移到原结点并放在最前/后面；
2. 将左/右兄弟结点的最后一个/第一个关键字上移插入到下移的元素的空位；
3. 左/右兄弟结点里的关键字全部后/前移一位。

◇ 若被删除关键字在终端结点，但是结点关键字个数删除后低于下限，**左右兄弟都不够借（兄弟合并，父亲下沉）**

1. 将原结点的父结点连接后的关键字插入到原结点关键字最后面。

2. 将原结点的左或右兄弟结点的关键字合并到原结点（前插或后插），并将连接也转移到原结点上。

3. 若父结点的关键字个数又不满于下限，则父结点同样要于与它的兄弟父结点进行合并，并不断重复这个过程。

4. 若父结点为空则删除父结点。

● $B+$树相比于B树：①内部节点只是索引，所有数据都保存在叶节点中；②叶节点之间通过链表连接，**可以进行范围查询**。

● 所有插入结点后需要进行调整的数据结构（堆、AVL 树、红黑树等）在插入节点后求其性质，**默认先进行调整操作**。

● 算法题通用

◇ solution n.解决方案；

◇ **次优解只扣两分，放心写**；

◇ 欲求某关系式的最大最小值，先把关系式化简；

◇ 算法题一定是基于数据结构的基本操作出的，做不出来的时候想想这种数据结构有什么基本操作；

◇ **【无关元素值只考虑树形**，王四-41】例，判断两棵树是否为镜像对称：若两棵树都是空树，则他们一定对称；若仅有一课是空树，则他们一定不对称。如此**递归判断**两棵树；

◇ **算法题的核心是分而治之**；

◇ 线性表（链表或数组）的长度为n，则其中间结点的索引为`(n+1)>>1;`；

◇ 若考 20 年算法题，所有所求数据可以变形成有关max和min的关系式均可以使用这种类似贪心的方法；

◇ 非算法题要求设计算法的一般使用伪代码；

● 算法题-树

◇ 后序遍历：求树高，求平衡度；

```
int leftHeight = getTreeHeight(root->left);
int rightHeight = getTreeHeight(root->right);
return max(leftHeight, rightHeight) + 1;
```

◇ 若树T使用孩子兄弟链表表示，则其向firstChild域递归时degree不变，向nextBro域递归时degree加一；

◇ 逆置链表

```
        while (curr) {
            ListNode* nextTemp = curr->next;
            curr->next = prev;  // 当前节点的next 指向前一个节点
            prev = curr;        // 更新前一个节点为当前节点
            curr = nextTemp;    // 移动到下一个节点
        }
        return prev;
```

- 冷门知识点

  ◇ 折半查找适用于 B+树；

  ◇ 散列表的装填因子：只给出了装填因子α，则此时平均查找长度为：$ASL=\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$；

- 外部排序：若内存中最多存放k条记录，则进行k路归并排序，可①使用**置换-选择排序**增大归并段长度来减少归并段个数；②使用**败者树**减少关键字比较次数、减少归并时间；③使用**最佳归并树**优化归并次序。

- 【败者树】：①在k个初始为升序（或降序）的序列中选择最小（或最大）元素，从而实现合并排序；②记录“冠军”的结点只能是**最小关键字所在段号**；③构建败者树时需要$n-1$次对比，从k个归并段选出一个最大或最小元素只需要对比关键字$\lceil\log_2 k\rceil$次；④败者树深度，k路归并排序表示**败者树有k个叶节点**；⑤败者树的构造，所有关键字所在段号视为叶节点，逐层比较后并冠军结点后更新败者树（即将原冠军结点对应叶节点的指针往后移并重新逐层比较）；

- 置换-选择排序：①在工作区内选择最大最小记录的过程需要使用败者树；②对于m个记录进行k路归并排序，最多可分为$\lceil\frac{m}{k}\rceil$组、故**最初会形成$\lceil\frac{m}{k}\rceil$个初始归并段**；

- 最佳归并树：使用的I/O次数为2WPL，WPL即为其**带权路径长度**。

- 栈和共享栈：二者均可能在栈空时出栈发生下溢在满栈时入栈发生上溢，但是共享栈中两个的空间比较灵活，所以**共享栈发生上溢的可能性较低**。

  ◇ 链栈不会溢出

- 共享栈栈满：`low.top-high.top==1`，此时**低位栈的栈顶指针高于高位栈的栈顶指针**。

- 卡特兰数：总方案数−不合法方案数。

$$Catalan_n=C_{2n}^{n}-C_{2n}^{n-1}=\frac{1}{n+1}C_{2n}^{n}$$

1. **括号匹配问题**：给定n对括号，有多少种合法的括号匹配方式。
2. **二叉树**：有n个节点的不同二叉树的数量。
3. **栈的出栈序列**：一个栈的进栈序列为$1,2,\ldots,n$，有多少种不同的出栈序列。

- 队列：**先进先出**。

◇ 队尾指针$rear$<adj.后方的，后面的>：一般指向**队尾元素**或队尾元素的**下一个位置**（数组编号增大的方向）。

▲ 指向**队尾元素**：先 rear++再赋值。

▲ 指向队尾元素的**下一个位置**：先赋值再 rear++

◇ 队头指针$front$<adj.前面的，正面的>：指向**队头元素**。

▲ 先来的人先打到饭，**出队**时先赋值再 front++。

◇ 若循环队列不采用【牺牲最后一个存储单元的方法】的方法，那么队空队满时均有 front==rear（即不能用作判断条件），而是采用另外增设的成员 length 或 tag 判断。

◇ 若循环队列采用【牺牲最后一个存储单元的方法】的方法：

▲ 队满条件(rear+1)%MAXSIZE==front。

▲ 队空条件 front==rear。

▲ 队列元素个数(rear+MAXSIZE-front)%MAXSIZE，其中 MAXSIZE 为数组大小。

◇ 在协调计算机部件间速度不匹配的问题（缓冲区）和多用户的资源竞争问题（排队，请求处理机）的问题上常使用队列解决。

● 后缀表达式：若栈顶运算符优先级**大于**当前运算符，先将栈顶所有优先级较大的运算符弹出，然后将当前运算符入栈。也可构造二叉树后进行**后序遍历**直接转换。

● 稀疏矩阵：只存储非零元素，压缩后就失去了随机存取的特性。非零元素采用【行标，列标，值】标识位置并存储。

◇ 三元组法：直接使用数组存储三元组。

◇ 十字链表法：每个结点除了使用【行标，列标，值】外，还有两个分别指向同一行和同一列的下一个非零元素的指针。

● 稀疏矩阵存储：总行数、总列数、三元组表。其中三元组表大小为非零元素个数乘表项大小，表项大小等于元素大小与所在行所在列数字的大小。

● KMP算法-next 数组

1. 先通过题目给定条件确定索引从 0 开始还是从-1 开始（这里以从-1 开始为例）并将模式串（记为T）依次编号，然后前两位 next 依次设置为-1,0；
2. 记A为【上一位字符】，k为【上一位 next 值】，比较A与T[k]进行比较，若相等则将当前 next 设置为$k+1$；

▲ 注意这里以及之后的所有比较都是拿A作为比较的基准。

3. 若不相等，则将k重置为上一步比较中T[k]对应的 next 值，再次比较直到相等或$k=0$；

4. 当$k=0$时，即与模式串首位字符比较时，若$A==\mathrm{T}[0]$成立则直接置当前 next 设置为$k+1=1$，若不成立则置当前 next 设置为0。

- KMP算法-nextval 数组、改良后的数组、修正后的 next 数组（**看清楚使用的是什么数组**）

◇ 写出 next 数组后，从低到高检查是否有 T[i]=T[next[i]]，若存在就将其 j=next[i]值设置为 next[j]。

- （下标从零开始）【nextval 数组值为-1】：**主串指针加一**，模式串指针归零。
- 树的带权路径长度WPL：树中所有**叶子**的带权路径长度之和称为树的带权路径长度。

1. **等于树中所有非叶节点的权值之和**；
2. 等于树中全部叶节点的带权路劲长度之和。

- ※※结点数=总度数（总边数）+1。最开始只有根节点，总度数（总边数）为零，之后每加一个节点，总度数和结点数同时加一。

◇ 注意区分树的度和总度数：树的度表示其最大孩子数，即 m 叉树。

- 对于任意一颗完全二叉树，$a_0=\left\lfloor\frac{n+1}{2}\right\rfloor$，$a_2=\left\lfloor\frac{n-1}{2}\right\rfloor$，其中$n$为节点个数。
- 对于任意一颗二叉树，若其叶结点数为n，则其所有结点数为$2n+1$。
- 将一棵满m叉树依次编号，对于编号为n的结点

◇ 其第k个孩子结点编号为$(n-1)\times m+1+k$，即**编号减一乘叉加一加 k**；

◇ 其双亲结点为$\left\lfloor\frac{n-2}{m}\right\rfloor+1$；

◇ 记住【**其倒数第二个孩子编号为mn**】即可。

- n个结点的二叉树不同的形态数量为其卡特兰数$\frac{1}{n+1}\mathrm{C}_{2n}^{n}$。

◇ 若对于$n+1$个结点的一般树转化为二叉树后，**其根节点一定没有右孩子**，故其转化后不同形态数等价于n个结点的二叉树不同的形态数量。

- 二叉树的**前序遍历和中序遍历对应入栈和出栈次序**。
- 层序遍历在同一层中**可以从前往后遍历也可以从后往前遍历**。
- 从中序线索树的最小结点，即最左结点，不断查找后继，一定能遍历完所有结点。
- 森林和二叉树：假设森林为F，树为T，转换而来的二叉树为B。

◇ B中无右孩子的结点数始终等于T/F中非终端结点数加一。

▲ T有n个结点，叶子结点个数为m，则B中无右孩子的结点个数为$n-m+1$个。

▲ F有n个非终端结点，则B中无右孩子的结点有$n+1$个。

◇ F有n条边、m个结点，则F包含T的个数为$m-n$。

- 后缀表达式：遇到运算符时，依次弹出栈中优先级**高于或等于**当前运算符的所有运算符并加入后缀表达式。
- 完全图边数：$C_n^2 = \frac{1}{2}n(n-1)$，其中$n$为结点个数。
- 图的连通就记三种特殊情况就行：环、最小生成树和完全图加边。
- 【有向图G的**强连通分量**】：①同一强连通分量中任意两点之间一定存在至少一条路径；②两相异强连通分量之间一定不存在两条不同路径，若存在一条路径则其一定为单向路径；③向有向图G中增加有向边，当且仅当**该边连接两相异强连通分量**且**使其之间形成双向路径**时，有向图G的强连通分量可能减少。
- 无向图中结点i的度为第i行所有有效元素之和，有向图则为第i行和第i列所有有效元素之和。
- **邻接矩阵是上、下三角矩阵并非图存在唯一拓扑序列的充要条件，其唯一性取决于具体结构。**
- 画哈夫曼树前先打草稿。
- M 叉（度为 M）哈夫曼树：若叶节点个数（已经添加虚段）为 n，则非叶节点数（合并次数）为$\frac{n-1}{m-1}$。
  - ◇ 设所求为x，则由结点数等于度数加一有$x+n=xm+1 \rightarrow x=\frac{n-1}{m-1}$。
- m 路归并排序每经过一趟排序剩下的记录数为原来的$\frac{1}{m}$。
  - ◇ 故若 n 条记录恰好进行 k 趟排序后完成，则$\frac{n}{m^k}<1<\frac{n}{m^{k+1}}$。
- 最小生成树：边数=顶点数-1。减去一条则不连通，增加一条则会出现回路。
- MST唯一性定理：若最小生成树没有使用无向图中相同权值的边，则MST唯一。
- 简单路径：一条**每个顶点只能出现一次**的路径。无环，无自指，环路不是简单路径。
- 直接插入排序和二分插入（二分用于寻找插入位置）一组，简单选择排序和堆排序一组（每次选取待排序序列中的最值直接追加放入有序序列。）
- 直接插入排序：从第二个元素开始，将每个元素插入到前面已排序的部分中。
  - ◇ 适用于链表，稳定的。元素序列**基本有序**的前提下，直接插入排序效率最高的；
  - ◇ 进行n躺后能保证前$n+1$个元素是有序的，但是不能保证其都在最终的位置上。
- 向量并行 CPU 可显著提高快速排序、归并排序、希尔排序的运行效率。
- 最后一趟排序结束前没有一个元素到达最终位置：希尔排序，归并排序，**基数排序**。
- 散列表的映射冲突-开放定址法
  - ◇ 线性探测法：$d_i=1,2,3,\cdots,m-1$。一定可以探测到散列表的每个位置。
  - ◇ 二次（平方）探测法：$di=1,-1,2^2,-2^2\cdots,\left(\frac{m}{2}\right)^2,-\left(\frac{m}{2}\right)^2$。

▲ 散列表长度m必须是一个可以表示为$4j+3$的素数才能探测到所有位置。

▲ 至少可以探测到散列表中一半的位置。

● 散列表的装填因子：α代表一个散列表中的满余情况，越大则查找效率越低。$\alpha=\frac{n}{m}$，其中n表示表中记录数，m表示散列表长度。

◇ 散列表的查找效率取决于三个因素：**散列函数、处理冲突的方法和装填因子**，与表长**无直接关系**；

◇ **α大于一当且仅当使用链地址法解决冲突**；

◇ 若只给出了装填因子α，则此时平均查找长度为：$ASL=\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$。

● 若待排序元素个数为n，每组元素个数为k并平分，则其使用基于比较的排序算法的时间下界为$O(n\log_2 k)$。

计算机组成原理

● 程序执行时间=指令条数× IPC × T。某措施对程序执行时间的影响都是通过公式中三个因数发挥作用。

◇ 优化数据通路可以使T减小。

● 【IEEE754浮点数特殊用途】：全0的-127表示**非规格化数**，全1的-128表示无穷大。

◇ $+\infty, -\infty$：尾数全为0，阶码全为1，正负由符号位决定。

◇ $+0, -0$：尾数全为0，阶码全为0，正负由符号位决定。

◇ 非规格化数：尾数非0，阶码全为0。

◇ 非法操作/非数值NaN（$Not\ a\ Number$）：尾数非0，阶码全为1。

● 【IEEE754浮点数规格化】：①原码规格化尾数最高**数值位**一定为1，补码规格化尾数符号位与最高数值位一定相反；②当基数为4时，原码规格化形式的尾数正数最高两位不全为0，负数最高两位不全为1；③当基数为8时，原码规格化形式的尾数正数最高3位不全为0，负数正数最高3位不全为1。

● 【IEEE754浮点数左右规】：①对阶和右规均会引起舍入；②左规即对尾数进行左移操作，阶码减小尾数增加，**仅在尾数小于一时进行（可能会进行多次）**；③右规即对尾数进行右移操作，阶码增加尾数减小，**仅在尾数大于二时进行（只需要一次）**。

● 浮点数计算时结果由尾数符号决定。

● IEEE754浮点数进行乘法运算的结果一定大于一，**一定不需要左规**。

● 【浮点数加减】：**看清楚题上给出的格式**，默认补码表示阶码尾数、**双符号位格式**。

1. 对阶：使两个数的阶码（**指数**）相等，**小阶右移向大阶看齐**，尾数每**右移**一位，阶码加 1。

   ▲ 若采用大阶码向小阶码看齐的原则，则尾数需要左移，最高有效位被移出，会导致结果出错；

   ▲ 若两个数阶码差大于 24，则直接取阶码较大的数为最终结果；

2. 【IEEE754浮点数规格化】：①原码规格化尾数最高**数值位**一定为1，补码规格化尾数符号位与最高数值位一定相反；②当基数为4时，原码规格化形式的尾数正数最高两位不全为0，负数最高两位不全为1；③当基数为8时，原码规格化形式的尾数正数最高3位不全为0，负数正数最高3位不全为1。

3. 【舍入】：①0 舍 1 入法/就近舍入法，即四舍五入；②恒置 0 法/截断法，**最简单的方法**。

4. 判溢出：仅对阶码判定溢出。

● 【※※※标志位】：只有进/借位标志位CF是“对于无符号数的整数运算有意义”。

◇ 零标志位ZF（$Zero\ Flag$）：判断当前数字是否为全0值。

▲ **无论是有符号数还是无符号数，ZF都有意义**。

◇ 符号标志位SF（$Symbol\ Flag$）：判断当前结果符号，$SF = F_{i=max}$，其中$F_{i=max}$为运算结果的最高位。

▲ **仅对于有符号整数运算有意义**。

◇ ※※溢出标志位OF（$Overflow\ Flag$）：表示带符号整数运算时结果发生溢出。**仅对于有符号整数运算有意义**。

1. 采用一位符号位，根据数值位和符号位的进位情况判断溢出：$OF = C_0 \oplus C_1$，其中C_0、C_1分别表示符号位和最高数值位产生的进位。
2. 采用一位符号位，正正得负或负负得正则为溢出：$OF = A_S B_S \overline{S_S} + \overline{A_S} \cdot \overline{B_S} S_S$。

◎ 这里**两个上划线不能连在一起**。

3. 采用双符号位，采用变形补码思想判断溢出：符号位`01`表示正溢出`10`表示负溢出，逻辑表达式为两个符号位进行或运算$OF = F_0 + F_1$。

◇ 进/借位标志位CF（$Carry\ Flag$）：表示无符号整数数加/减运算时的进位/借位（溢出），$CF = C_{out} \oplus Sub$，其中Sub为加减法控制信号，加法0减法1，C_{out}即为最高位产生的进位。

▲ **仅对于无符号数的整数运算有意义**。

● $A - B$后使用标志位判断AB大小：$\mathrm{ZF} = 0$时结果为零，$A = B$，下面讨论其他情况。

◇ CF和ZF判断无符号数：$\mathrm{CF} = 1$时结果溢出，$A < B$；$\mathrm{ZF} = \mathrm{CF} = 0$时，$A > B$；

◇ OF、SF和ZF判断有符号数：$\mathrm{OF} \oplus \mathrm{SF} = 1$时，$A < B$；$\mathrm{OF} \oplus \mathrm{SF} = 0$时，$A > B$；

● 【无损转换】：$char \rightarrow int \rightarrow long$（32位）$\rightarrow double$、$float \rightarrow double$。

● 【精度丢失】：①64位$long \rightarrow double$可能会有精度损失，$double$尾数长度为53位，多余的部分会丢失，但是不多于53位就是无损转换；②$float \rightarrow int$，可能会溢出（$float$表示范围更大）及损失精度导致$float$的小数部分会丢失，但是$float$**是整数就是无损转换**；③$int \rightarrow float$可能会损失精度，$float$尾数长度为23位，同上。

● MSB**为最高有效位，**LSB**为最低有效位**。

● **类型转换优先级大于加减乘除**。

● 【交叉编址多模块存储器】：有轮流启动和同时启动两种方式，若存储器位数×体数等于总线位宽则一定是**同时启动**，此时总线频率等于存储器频率。

● 【DRAM芯片的行缓冲计算】：每行的列数×每个存储单元的位数，其中对于$A \times B$位的芯片，每行的列数为$\sqrt{A}$、每个存储单元的位数为即为B（注意若**其为位拓展的芯片应当选择组成其的基本芯片**）。

- 【固态硬盘SSD】：①属于电可擦除ROM即EEROM，使用Flash芯片**可进行多次快速擦除重写**；②以页Page为单位读/写，以块Block为单位"擦除"；③由于Flash需要先擦除再写入，因此**写速度比读速度要慢**；④进行写操作时必须按**块内页**的顺序写入；⑤其中的闪存翻译层和磁盘控制器一样均用于地址翻译。

Cache&TLB专题

- 【TLB、Page缺失（缺页异常）】：①TLB缺失后会访问**内存**中的Page（访存），若Page中存在页表项则将其加载到TLB且使用更新后的物理地址继续操作；②若Page中不存在页表项或失效则触发缺页异常，从**磁盘或交换区**加载其到TLB和Page并使用更新后的物理地址继续操作；
- 内容寻址存储器CAM：TLB和Cache。
- 【Cache】：①提到缓存寄存器默认等于Cache；②平均访存时间= $Cache$存取时间+$Cache$缺失率×主存存取时间；③Cache缺失引起的时间开销：主存延、层次结构即多级Cache、替换算法；Cache行中的修改位一般指的是回写法的脏位，与LRU替换算法无关；
- 在正常情况下，**查询TLB和查询页表**通常是连续发生的，均为使用虚拟地址查找物理地址的过程；不与Cache连续，查询 Cache 时已经查询到物理地址需要查找数据。
  ◇ 是TLB页表的快速缓冲，Cache是主存的快速缓冲。
- TLB缺失后有可能直接**在 Cache 中找到页表内容**。页表存储在主存中，Cache保存主存的副本。
- Cache缺失通常由**硬件**处理，TLB缺失可以由 **硬件或软件**处理。
- 【Cache完全由硬件实现】：①对所有程序员透明；②Cache缺失时由硬件进行自动处理，一般不需要切换进程。
- 【三种Cache映射策略】：①直接映射不需要算法位；②全相联映射不需要Cache行号/组号；③三种策略均会出现『刚被替换出的数据又被访问』的情况（**会导致Cache命中率为零**），且其中直接映射发生的概率最高。
- 【回写法和全写法】：回写法加信息位，全部改完了再写回主存；全写法**加访存次数**，把数据**同时写入Cache和主存**。
- Cache缺失率
  1. 明确一条指令会访问几次数据和总共会访问多少次；
     ▲ 对于类似 `a[k]=a[k]+1;`的代码会连续访问两次；
  2. 明确一个Cache块一共能存放多少个数据、第一个数据是否在整块的开头；
  3. 仅直接映射或全相联映射在循环条件下可直接对单个块的访问进行分析；

- 指令流水线中，TLB缺失和Cache缺失均只可能在取指IF段和访存Mem段出现，与写回WB段无关。
- 【L1 Cache分为指令Cache和数据Cache】：求组数时正常的算一个再乘二（所以一般答案都是偶数）。
  - ◇ 计算组号时应该将当个Cache行数÷关联度作为组数，相当于L1 Cache已经天然分了一组了，但是计算总容量的时候应该把另一个加上。
- 【Cache每行的控制部分】：不加数据部分的总大小，即TAG +替换算法位+写回法的修改位+有效位。
- 【k路组相联映射的Cache缺失率】：①$k > 2$时一般均为两个连续的块放在两个连续的组中，轮转一圈后再会放到同一组中；②务必分析**【是否存在块置换？】**、**【是否存在相互替换？】**。
- Cache**关联度**：一个主存块有可能映射到几个Cache行，对于k路组相联的关联度为k。
  - ◇ 关联度决定：①比较器个数；②LRU替换算法需要的额外的位数（与总行数和总组数无关）；
  - ◇ 关联度可以（**但不一定**）降低Cache缺失率。

- 【总线带宽和某种总线事务的数据传输速率】
  - ◇ 【总线带宽】：总线在单位时间内能够传输的**最大数据量**，计算方法为**总线带宽=总线位宽×总线工作频率**，其中【上升沿和下降沿各传送一次数据】即为**工作频率等于总线时钟频率的两倍**。
  - ◇ 【某总线事务的数据传输速率】：与总线事务的具体行为密切相关，需要根据题给的具体动作计算。
- 【三种I/O控制方式】：①程序查询I/O方式**不会造成中断**；②程序中断I/O，CPU会在每个**指令周期的末尾**检查中断、不适用于低速外设；③DMA方式高速外设低速外设均适用**但不适用于键盘和鼠标**（其要求CPU能够立刻响应，而DMA不过CPU）、不用保护和恢复现场（没有中断）。
- 【DMA传输的流程】：①接受请求后，初始化控制器并启动磁盘，其中由初始化主要为设备驱动程序设置**内存地址寄存器**MAR（目标地址）、**数据计数器**DC（数据块大小）初值和**传送长度计数器**WC（剩余传输字节数）；②传输一块数据块到**数据缓冲寄存器**DR；③发送总线请求；④由DMA直接控制总线传输；⑤“DMA结束”中断，由中断服务程序完成。
  - ◇ **【每次传送一个字，传送整个数据块】**：I/O接口中的数据缓冲区（数据缓冲寄存器）每充满一次，DMA控制器就需要发出一次总线请求；
  - ◇ CPU在每个**存储周期**（总线事务）结束后检查是否有DMA请求，DMA**请求优先级默认最大**；
- 【DMA冲突处理】：①停止CPU访问主存；②周期挪用，CPU与DMA均有访存（同时访存时DMA优先度更高，但CPU先访存时需**等到存取周期结束后**）、周期挪用指主存的**存取周期**；③CPU与DMA

交替访存，一个CPU分成两个用、不需要总线使用权的申请建立和归还过程、适用于 CPU 的工作周期比主存存取周期长的情况。

- 【结构冒险\资源冲突】：①指令、指令Cache；②**硬件阻塞**或**软件**空指令NOP。
- 【数据冒险】：①**硬件阻塞**或**软件**空指令NOP；②数据旁路；③编译优化。
  - ◇ 空指令NOP同样有完整的五个阶段。
  - ◇ **数据相关**在**循环**中可能后面相关前面。
- 【控制冒险】：①**分支预测**；②提前形成条件码；③编译优化；④指令预取。
- 【分支预测】：尽早判别转移是否发生，尽早生成转移目标地址。①简单预测，永远猜True或False；②动态预测，根据历史情况动态调整，有较高的预测准确率。
- 【存储器预取】：①DDR3/4使用8位预取技术、DDR2使用4位预取技术；②通过芯片中的I/O缓冲区实现。
- 【定点数乘法运算】：①尽量直接转换成真值计算；②实现N位（**不包括符号位**）补码一位乘时，乘积为$2N+1$位；③原码、补码一位乘法中，最多均需要N次移位，$N+1$次加法运算；④原码、补码乘法的符号位都需要**单独进行异或操作**；⑤为了加快运算会有**辅助电路**实现$(-x)$的补码的运算；
- 【$Booth$算法的移位法则】：记y_n为MQ最低位，y_{n+1}为辅助位，则若$y_n \oplus y_{n+1}=0$则部分积右移一位，否则先加或减（y_n为一即为加否则为减）当前运算值。
- 【补码乘法溢出判断】：①若用$2n$位保存最终乘积，则不会溢出；②若用$2n$位保存中间结果，n位保存最终乘积，则**高$n+1$位相同时不溢出**（对于无符号数乘法当且仅当高n位全0时才不溢出、对于有符号补码乘法，当且仅当前$n+1$位数值全相同时不溢出）。
- 【补码加减交替法/不恢复余数法】：①符号位参与运算、所有数均采用双符号位；②若字长为$n+1$，则只用左移n次，上商$n+1$次，最后一次上商余数不左移；

- **！！！所有需要累加的数据（时间复杂度、地址计算时）好好想清楚从零开始还是从一开始！！！**
- **补码没有负零**，8000 0000H表示的是-2^{31}，不是零。
- 计算机内特殊部件
  - ◇ 阵列乘法器：所有部分积同时产生并组成一个阵列，可在一个时钟周期内完成乘法运算；
  - ◇ 内存管理单元MMU：主要是**地址转换**；
  - ◇ 栈区$stack$：由编译器自动分配释放，实现函数调用，用于存储**函数传递的参数、局部变量**等，从用户空间的高地址向低地址增长；
    - ▲ 入栈：$SP \leftarrow (SP)-'1'$；

◇ 堆区$heap$：一般由程序员分配释放，程序结束时可能由 OS 回收，和堆排序的堆不是一个东西，分配方式类似于链表；

◇ 浮点字寄存器：位数只由系统中所能表示的最大的浮点数决定；

◇ 【虚拟存储器】：容量记为计算机逻辑地址空间的大小，由逻辑地址位数决定，**与主存辅存无关**；

◇ 【控制存储器CM】：①在 CPU 中，用于存储微指令，其容量为$2^N \times M\ bit$（N为下地址字段长度、M为微指令字长）；②**按**微指令**地址访问**；③机器运行时**只读不写**；

◇ 【重定位寄存器=**基址寄存器**】：存放当前进程**起始物理地址**；

◇ 【界地址寄存器】：存放当前进程**最大逻辑地址**；

◇ 【多路选择器MUX】：根据控制信号从多个输入中选取输出，一般用于在同一个数据通路中实现多指令。故MUX的输入信号对应于该数据通路能执行的指令；

● 某部件是否透明

◇ 对所有用户：程序计数器PC，通用寄存器组X，累加寄存器ACC；

◇ 对汇编程序员可见：中断字寄存器、基址寄存器/变址寄存器、程序状态字寄存器PSW；

◇ 完全透明：指令寄存器IR、微指令相关、高速缓存$Cache$、MAR和MDR、暂存寄存器R。

◇ 运算器的大部分部件，包括移位器、乘法器、先行进位链，均为透明。

● 硬件和软件

◇ Cache 替换使用硬件实现，主存替换使用软件实现；

◇ 磁盘控制器和磁盘驱动器都是硬件，磁盘驱动（程序）以及所有设备驱动都是软件；

● **冯诺依曼计算机通过程序计数器PC，即控制器，依据不同的阶段区分数据和指令**。

● **【预处理、编译、汇编、链接】**。

● 翻译程序将高级语言源程序转换为机器语言程序，分为编译程序（一次性翻译成可执行文件）和解释程序（边解释边执行）。

● 数据按边界对齐

◇ `char` 类型占用 1 字节，起始地址可以是任意地址。

◇ `short` 占用 2 字节，`int` 占用 4 字节，`double` 占用 8 字节，起始地址必须在响应的倍数地址上。

● 【N位补码表示范围】：$[-2^{N-1} \sim 2^{N-1}-1]$，其中记正数最大值为$n-1$位一，即$2^0+2^1+\cdots+2^{N-2}=2^{N-1}-1$。

● 若题上没有明确给出大端还是小端，则说明不论是哪一种，答案都是固定的。

● 从 32 位逻辑地址 LA 中提取页号 P 等：`(((unsigned int) (LA)) >> N) & M`。其中 N 为提取的 P 在多少位，M 为与 P 等长的 1 构成的字段，通常使用十六进制表示。

- 算术移位和逻辑移位：**仅算术右移补符号位**，其他（逻辑右移、逻辑左移、算术左移）全补 0。
- 无符号的移位运算使用的是逻辑移位，有符号数的移位运算使用的是算术移位。
- 正数加负数一定不会溢出。
- 无符号转有符号：注意有符号数需要根据最高位确定符号，**一般为负**。
- 【为什么无需对无符号数进行溢出判断】：**无符号数常用于地址运算，因超出表示范围而产生的截断等价于取模运算**。
- 磁盘阵列$RAID$：冗余、纠错（海明码）、校验（位/块交叉奇偶校验）、**可以并行工作改善性能**。
- 现代内存可以实现在同一时钟周期内在上升沿和下降沿各传送一次数据。
- 现代磁盘采用固定位密度，外围磁道的信息量大于内圈。
- 偏移寻址
  - ◇ 【基址寻址】：起点为程序的起始存放地址；基址寄存器内容由操作系统管理，用户只可选择基址寄存器；有利于多道程序并发运行，可用于**编制浮动程序**；
  - ◇ 【变址寻址】：**起点由用户决定**；适用于**数组处理，编制循环程序**；
  - ◇ 【相对寻址】：起点为$(PC)+'1'$；适用于**转移指令**；
  - ◇ 【堆栈寻址】：①含有堆栈指针SP的均为堆栈寻址；②SP指向的是寄存器则为硬堆栈、向的是主存则为软堆栈。
- 指令执行过程中，基址寄存器的内容不变，变址寄存器的内容可能改变。
  - ◇ 基址寄存器内容由操作系统决定，变址寄存器可由用户操作。
- 除相对寻址偏移量（补码）之外的所有地址都是无符号整数。
- 单周期 CPU：一个时钟周期完成一条指令，时钟周期的时间以执行时间最长的指令为主。CPI 为 1。控制信号不变，除 PC 外寄存器值不变。采用多总线，单总线会冲突。
- 操作数只可能存在于寄存器和主存单元中。
- M 体低位交叉存储器：若一个存储单元为 `k bit`，则其在一个连续传输的存储周期内可总共提供 `k*M bit`。
- 【设计芯片求所用数量，根据地址求容量】：某按字节编址的计算机有$4000\text{H} \sim 5\text{FFFH}$，则一共$5\text{FFFH}-4000\text{H}+1=2000\text{H}$个存储单元，即$0\times16^0+0\times16^1+0\times16^2+2\times16^3=2^{13}\text{B}$。
  - ◇ 注意从0开始。
- 主存-外存的映射只能使用全相联映射。
- 指令格式的题看清楚是求有效地址还是操作数。
- 在循环中，标号 `exit` 表示跳出循环通常指向循环后第一条指令地址，标号 `loop` 表示进入循环通常指向循环的第一条指令地址（如果是 `for` 循环指的则是赋初值后的那条指令的地址）。

- 跳转指令的跳转目的地址计算：当前指令的下一条指令地址+指令字长× OFFSET。
- SignExt 部件：符号拓展。输出位数大于输入位数的均为拓展部件。
- CPU中仅通用寄存器组能够存储数据，所以类似【CPU与寄存器之间进行数据交换】的说法均是错误的。
- 单周期 CPU 与多周期 CPU：①单周期 CPU 无需使能信号，依据时钟信号每时钟周期执行一条指令，执行过程中单周期 CPU 的**控制信号不变**、**每个部件只能使用一次**（多周期 CPU 的控制信号会发生改变且部件可重复使用）；②单周期 CPU 的CPI始终为一，始终小于多周期 CPU 的CPI；③单周期 CPU 的时钟周期始终大于多周期 CPU；
- 速度差距较大的两设备间可采用：①异步传输；②缓存，**Buffer** 或 **Cache** 均可；③DMA。
- 【流水线性能】：①最大吞吐量，最理想情况下与每个流水段耗时成反比；②加速比，最理想情况下等于流水线段数（超标量流水线的原理）。
- 【指令集ISA规定内容】：①包含指令相关，指令格式、操作数、操作类型；②包含硬件可操作的基本功能，寄存器数量、数据类型；③包含异常和中断的处理方式，如缺页、溢出等；④**不涉及实现细节**，如流水线设计、缓存大小等；⑤**不涉及硬件平台**，如核心数、时钟频率。
- 定长指令字长度固定可以直接进行PC加一，变长指令字则需要通过专门的PC**增量器**进行计算。
- 微程序通常以存储器ROM形式存在，**微程序入口地址由机器指令的操作码字段形成**，一条机器指令对应一个微程序。
- 一般情况下，微程序固定执行一个时钟周期。
- μPC接受微地址→控制存储器CM输出微指令→存储于μIR。
- 【控制存储器CM】：①在 CPU 中，用于存储微指令，其容量为$2^N \times M\ bit$（N为下地址字段长度、M为微指令字长）；②**按**微指令**地址访问**；③机器运行时**只读不写**。
- **【水平型微指令、下地址字段和控制存储器CM容量】**
  - ◇ 操作控制字段：①直接编码方式，微命令数等于操作控制字段位数；②字段直接编码方式，对于每个互斥组，n位操作控制字段可表示$2^n - 1$个微命令；
  - ◇ 判别测试字段：①直接编码方式，外部条件等于其位数；②字段直接编码方式，对于N个外部条件，需要n位表示，其中$2^n \geq N + 1$，加1是因为还有**无条件转移**的情况；
  - ◇ 下地址字段：微指令字长−操作控制字段长度−判别测试字段长度即可，与控制存储器CM的容量（可以存储多少微指令）相等。
- 【水平型微指令和垂直型微指令】：水平型并行能力强、执行时间短，但微指令字长较长；垂直型的微指令字长较短，微程序长度较长。
- 微指令最大总数由微指令中下地址字段的最大位数决定。

◇ 微指令编码分为操作控制字段和下地址字段；

◇ 字段直接编码的微指令长度更短，控制存储器CM利用率更高，但不如直接控制法快；

◇ 垂直型微指令相比于水平型包含的微命令更少、指令更短、微程序更长。

- $CISC$繁杂：指令长度不固定，采用微程序控制，指令执行时间较大。
- $RISC$精简、速度快：指令长度固定，只有 Load/Store（取数/存数）指令访存，硬布线控制，一定采用指令流水线技术，通用寄存器的数量相当多。
- 流水线周期：最长耗时+寄存器延时（锁存器）。
- 【数据通路】：①IF段，程序计数器PC、指令存储器和下地址的计算逻辑；②ID段，操作控制器、取操作数逻辑和立即数符号扩展模块；③EX段，算术逻辑单元ALU和分支地址计算模块；④MEM段，数据存储器读写模块；⑤Wb段，寄存器写入控制模块。
- 流水寄存器保存信息：**后面流水段**需要用到的所有**数据信息**、**前面传递来**要用到的所有**控制信号**。
- 流水线的控制信号：①控制信号在ID段中的指令译码器中产生；②取指IF段和译码ID段所有指令动作一致，不需要控制信号；③每到达一个时钟信号CLK，控制信号即往后传送一段，而**不是统一传送**；
- 指令流水线中，一般指令均会向最长指令对齐，故部分流水段设置了 nop 操作。
- 采用 n 个处理机所获得的加速比不一定是一个处理机的 n 倍，需要额外资源进行协调。
- 【LOAD · USE数据冒险】：采用转发后额外需要一个时钟周期的阻塞，不采用转发需要两条NOP指令。
- 高级流水线

◇ **超标量流水线**技术/动态**多发射技术**：每个时钟周期内可并发多条独立指令，$CPI < 1$；

◇ 超长指令字技术/静态多发射技术：增加流水线级数使更多的指令同时在流水线中并行执行，$CPI < 1$；

◇ 超流水技术：时钟周期内再分段，**提高流水线主频**的方式来提升流水线性能，$CPI = 1.$

- 超标量流水线的 CPU 必须设置多个不同的功能部件，利用部件的并行性提高并发度。
- 若指令流水线中存在根据情况决定是否跳转的指令 JMP、JC，则在没有分支预测的情况下，默认存在三个时钟周期的阻塞使当前指令的 ID 段在上一条指令的 WB 之后。
- 解决指令流水线冲突的阻塞是**硬件阻塞**，插入空指令属于软件，二者不一样。
- 定长指令字的 PC 可以直接+‘1’，而变长指令字的 PC 需要设置专门的增量器辅助。
- 多指令流单数据流$MISD(Multiple\ Instruction\ Single\ Data)$：不存在，多指令流必定多数据流。
- 多线程系统的特长需要**任务可分割为多个可并行的小任务**才能发挥完全特长。

● 仅并行向量处理系统可以确定是多 CPU 系统，其他的多线程系统、多进程系统、实时系统均不一定是多 CPU 系统。

● 【多处理器系统】：①也叫共享存储系统MIMD，**所有主存储器均属于共享的单一地址空间**，通过存取指令来访问系统中的所有存储器；②分为统一访存UMA和非统一访存NUMA，前者访存时间与哪个处理器提出访存请求及访问哪个字无关，后者访存时间则**取决于哪个处理器提出了访问请求以及访问哪个字**（主存被分割并分配给了同一机器上的不同处理器或内存控制器）。

● 多指令流多数据流$MIMD$中，

◇ 消息传递类型$MIMD$不能通过存取指令访问不同节点的私有存储器，而是使用消息传递进行数据传送；

◇ 共享存储$MIMD$/多处理器系统SMP：具有共享的单一地址空间，通过存取指令来访问系统中的所有存储器。

● 【$MISD$不存在】：其中M表示【复杂、多个】、S表示【单个】、I和D分别表示【指令】和【数据流】，所以$MISD$不存在（不存在多条指令同时对单一数据流进行操作）。

● I/O接口：①**选址**功能；②传送命令功能；③传送数据功能；④反映 I/O 设备工作状态的功能。

● I/O指令：①执行时，CPU 使用地址总线选择I/O端口使用数据总线在 CPU 通用寄存器和I/O接口之间进行数据传送；②仅独立编址方式才有I/O指令。

● 地址变换中可能发生的中断或异常有：缺页异常、越界和访问权限错误。

● 内中断/**异常**（有关）和外中断/**中断**（无关）的最大区别是**与当前执行的命令是否有关**。

● **内部异常**发生在指令执行过程中，响应时机是立即中止指令；**外部中断**发生在指令执行完成后，响应时机是指令周期执行完毕时。

● 【缺页异常】：①仅发生在**页表查询**阶段（MMU进行地址转换时）；②异常处理后需要重新访问 TLB；③可以认为**仅缺页异常会在处理完毕后返回当前指令重新执行**。④可以认为**缺页异常**是唯一的可能产生多次的（广义）中断。

● 【缺页异常的流程】：①MMU生成缺页异常信号；②检查页面是否在外存和交换区、并将其调入内存（依据情况进行页面置换）；③更新页表并将对应有效位置一。

● 【缺页异常的优化】：①页面置换算法；②降低缺页率；③降低磁盘（默认外存）存取时间。

● 【缺页率】：工作集大小、物理内存大小、页面置换算法、页面大小、进程的局部性和数量、TLB 的性能、页面预取性能。

● 中断隐指令保存断点（PC内容），中断服务程序保存现场（用户可见的寄存器的内容）。\

● 【时间片轮转】：①时间片用完会触发时钟中断，将进程由运行态转换到就绪态；②时间片轮转实现了对于 CPU 的虚拟化；③**进程主动让出处理机等价于进程时间片用完**。

- 外部中断的断点为下一条指令，缺页缺段（故障-内部异常）的断点为当前指令，非法操作码和除数为 0 不能回到断点执行。
- 【突发传输】：可以显著提高连续数据的传输速率，但对离散数据的提升不明显；
- 【同步总线】：时钟频率不一定等于工作频率，例如可能在上升和下降沿各传输一次数据。
- CPU 响应外部中断的条件：①中断源有中断请求；②CPU 允许开中断，不可屏蔽中断除外；③在每条指令结束后的中断周期响应。
- **响应中断=中断隐指令**，处理中断=中断服务程序。
- 中断服务程序的最后一条指令是中断返回程序，其不完全等于无条件转移指令或普通返回指令。
  - ◇ 中断服务程序比无条件转移指令多了：恢复现场、恢复程序计数器PC和程序状态字寄存器PSW；
  - ◇ 中断服务程序比普通返回指令多了：恢复程序计数器PC（不恢复程序状态字寄存器PSW）。
- 无条件转移指令J、JMP指令可以直接指定绝对跳转地址、**其后跟地址可能即为其目标地址**。
  - ◇ 条件转移指令JE和条件分支指令BE、bne等均一定为相对寻址。
- 中断处理结束和进程阻塞均会导致进程的调度。
- 中断响应：由 CPU（硬件）在中断响应周期自动完成。关中断，保存断点和程序状态字寄存器。
- 中断处理：由**中断服务程序**即软件完成。保存现场和中断屏蔽字、**在开中断的状态下进行中断事件处理**、恢复现场和中断屏蔽字、中断返回。
- 【检测溢出的自陷指令】：使 CPU 自动查询OF溢出标志位，在其有效时转异常处理。
- 时钟中断和网络传输到达的数据包都是外部中断。
- 内部异常由 CPU 内部检测，发生异常时立即响应。
- **中断响应**优先级由硬件或查询程序的查询顺序决定不可动态变化；中断屏蔽字可以确定**中断处理**优先级可动态变化。
- 中断屏蔽字：1 表示**屏蔽该中断源的请求**，0 表示可以正常申请。自己不可以中断自己，自己那位始终为 1。
  - ◇ 其中**屏蔽该中断源的请求**表示若$A > B$（中断处理），则A先到达且已经在**广义中断处理**时，无论B的中断响应的优先级如何，CPU均无法检测到B的中断请求。
- 广义中断处理中，【由硬件处理】表示中断响应阶段、【由软件处理】表示中断处理阶段。
- **中断屏蔽字决定中断处理及其处理完毕的次序**。
- 中断响应持续时间较短，若对于两个中断请求AB，中断响应优先级$A > B$，中断处理优先级$B > A$，则开启多种中断的情况下，A中断响应完成后正在中断处理时会被进入中断处理的B打断。
- 中断驱动I/O可用于处理随机事件，例如用户突然按下键盘上的某个键等。

- 采用中断驱动I/O控制打印输出时，CPU 与打印机直接交换打印信息，不会交换主存地址；使用 DMA 方式时 DMA 控制器可以直接读取主存中的数据块。
- I/O接口的三根线：不是**信号**都是数据线传的。
  1. 数据线：数据缓冲寄存器，**状态/控制寄存器**（接口和设备的状态**信息**、*CPU*对外设的控制**命令**）；
  2. 地址线：要访问的I/O接口中的寄存器的**地址**和读/写控制**信号**；
  3. 控制线：读/写**信号**，仲裁**信号**和握手**信号**。
- 总线上，只有数据信号是双向传输的。地址、控制、状态均为单向传输。
- CPU 和主存均通过I/O总线与I/O接口相连，I/O接口通过通信总线与外设相连。
- 计算机内栈顶指针一般从高位到低位入栈。

操作系统

- semaphore *n.*信号量、progress *n.*进程。
- 【不同**进程**之间的资源共享】：①私有的，即**打开文件表**、**堆栈指针**、进程的全局变量和静态变量、**页表指针**；②共享的，共享内存（用于进程间通信）、信号量和消息队列、设备。
- 【父子进程】：可以共享一部分资源，但是**不能共享虚拟地址空间**。因为是逻辑地址且其有不同的PCB用于区分进程。
- 【同一进程下，各个**线程**之间】：①私有的，即执行上下文（包括程序计数器、**堆栈指针**等）、寄存器；②共享的，即进程的**虚拟地址空间**、代码段数据段、**全局变量**和堆，打开文件表，信号量机构（**互斥锁**、条件变量）等同步机制。
  - ◇ 不同线程均需要访问原进程的源代码（进程=动起来的程序），所以应当**共享地址空间**；而每个线程需要进行自己的线程调用，所以堆栈指针（执行上下文）等私有。
  - ◇ 在支持**硬件多线程**的计算机中，每个线程都拥有独立的通用寄存器组和程序计数器。
- 【**系统调用流程**】：①传递参数；②使用软中断指令即自陷指令；③**软中断指令的中断响应阶段**，即保存断点和PSW；④**硬件操作切换到内核态**并查找中断向量表以跳转到系统调用处理程序；⑤**软中断指令的中断处理阶段**，保存现场等，后略。
- 【进程同步】：①**仅纪录型（带等待队列）的信号量完全遵守了四条互斥访问准则**，整数型信号量、Peterson算法、互斥锁均不满足让权等待；②`wait`、`signal` 操作可以解决一切同步问题，但无法防止系统死锁。
- 【判断算法是否能够保证互斥是否会导致饥饿】：首先将所有可能的进程的执行代码全写出来并标号，**若存在一个执行序列能使多个进程进入临界区说明该算法不能保证互斥**；而若进程地位平等，一般可以认为不会导致饥饿。
  - ◇ `while(FLAG);`中若 `FLAG` 为真会一直开在这里，即若想通过应该使 `FLAG` 为假才行。
- 【处理死锁】：①预防死锁（破坏死锁条件）、避免死锁（银行家算法）、检测死锁（进程资源图）；②**限制申请资源顺序**统一理解为按序号顺序递增，属于预防死锁；③进程资源图用于**检测死锁**，在每个资源均为单实例时使用拓扑排序检测若出现环路说明**可能发生死锁**否则**一定不会发生死锁**，非单实例时不能简单的通过有无环判断。
- 【管程】：①管程通过条件变量来实现阻塞进程，条件变量**【没有值】**，调用这两个操作时**都不用判断条件**；②只能通过调用管程内的过程访问**共享数据，无法访问私有数据**；③同一时刻内管程内函数只能被一个进程使用；④能实现同步和互斥。

- 【进程通信】：①低级通信方式是PV操作，高级通信方式有共享存储、信息传递、管道通信（还有共享文件等）；②共享存储，可以分别基于数据结构（只能存放固定数据、是低级通信方式）和存储区（存放位置由进程控制、高级通信方式）实现；③信息传递，通过操作系统提供的原语实现，可以直接发送到接受进程的信息缓冲队列上（直接通信）也可以先发送到中间实体信箱中（间接通信）；④管道通信，即内存中开辟一个大小固定的缓冲区。
- 【管道通信】：①管道只能**半双工通信**；②读写都可能被堵塞，如果没有**写满**则不允许读、如果没有**读空**则不允许写；③一个管道允许多个写进程一个读进程。
- 【用户级线程】：①**对操作系统透明**；②线程切换不需要转换到内核空间，开销较小；③某线程阻塞时，**进程内的所有线程都被阻塞**（罪魁祸首）；④由用户空间的线程库实现，可以在不支持内核线程的系统中实现。
- 【多对一模型】：①多个用户级线程映射到一个内核级线程；②**一个线程被阻塞后与该进程中所有线程也都被阻塞**；③一对一模型和多对多模型不会发生以上情况。
- 【程序运行时的内存映像】：①一般包含代码段（只读、可共享）、数据段（包括全局变量和静态变量）、进程控制块PCB、堆、栈；②堆用于存放动态分配的变量，通过调用 `malloc` 函数动态地**从低地址向高地址**分配空间；③栈用于实现函数调用，**从最大地址**往低地址方向增长；
- 【虚拟内存】：①基于**离散分配**（和请求机制）实现，动态分区分配的连续分配方式不能用于虚拟内存；②最大容量等于逻辑地址空间的大小、同时也等于虚拟存储器的容量，实际容量等于主存辅存的容量之和；③**进程的虚拟地址空间的大小**仅由虚拟地址空间的位数决定；④主要特征为虚拟性。
  - ◇ 虚拟内存≠虚拟存储器；
  - ◇ 【虚拟内存的实现】：离散分配、请求机制、内存和外存、页表或段表、中断机构、地址变换机构。
- I/O软件层次结构：用户层软件，设备独立性软件，设备驱动程序，中断处理程序。
  - ◇ 设备独立性软件：**实现系统调用**。与设备的硬件特性无关的功能几乎都在这一层实现。
  - ◇ 设备驱动程序：控制I/O设备工作，执行$read$和$write$命令、**计算柱面号、磁头号和扇区号**、设置设备寄存器、检查设备状态等。凡是包含物理设备、**设备操作**的功能都在这一层实现。
- 【柱面号、磁头号和扇区号】
- 【**设备驱动程序**】：①与硬件平台、操作系统、I/O控制方式有关；②其中的一部分必须用**汇编语言**书写，**部分内容已固化在ROM中**；③允许可重入，一个正在运行的驱动程序通常会在一次调用完成前被再次调用；④驱动程序**不允许系统调用**。

● 【设备驱动程序的处理过程】：①将抽象要求转换成具体要求，如将read命令中的盘块号按地址寄存器的格式转化为盘面号、磁道号和扇区号；②检验权限、检查设备状态；③传递参数，如将磁盘地址、内存地址和传送字节数等传送到设备控制器的寄存器中；④启动I/O。

● 【打开文件前后使用的参数不同】：①open系统调用，文件存放路径、文件名、要对文件的操作类型；②read系统调用，文件描述符、缓冲区首址、传输字节数，**不需要文件名和存放路径**。

● 【※※※**成组链接法**】：用来存放一组**空闲盘块号**的盘块称为**成组链块**，每一组成组链块的最后一个（指**最后一个进行分配的**）空闲盘块作为成组链块，用于保存另一组空闲盘块号，每个成组链块使用**空闲盘块计数**表示当前组中空闲的盘块数量。

◇ 分配时，对于每一个请求的盘块，分配给用户后依次将指针下移一格；若该指针指向的是最后一个盘块（成组链块），则直接在下一组中选取盘块进行分配，**当前组中指向下一组的空闲盘块不参与分配**；

◇ 回收时，指针上移一格，若当前成组链块已满还有一个盘块进行回收，则将现有已记录 n 个空闲盘块号的成组链块号记入新回收的盘块，**将当前回收的盘块作为新的成组链块**。

● 【文件系统】：①文件系统中利用**目录**组织大量文件；②为了允许不同用户使用相同文件名，文件系统采用多级目录；③文件系统中每个目录项都是FCB，UNIX系统中的目录项则为文件名和索引节点指针。

● 【内存映射文件】：①可以将文件的内容直接映射到进程的虚拟内存中，使得文件的访问就像对内存的访问一样；②创建时，使用open系统调用和mmap系统调用，返回可以用于后续内存操作的指针；③访问时，通过操作**内存映射区域**来访问文件内容，进程可以直接读取和写入内存映射区域**而不需要read系统调用和write系统调用**，且由于内存映射区域与文件内容直接关联，**对映射区域的修改也会直接反映到文件中**；④**多个进程可以映射同一个文件**以实现共享；

● 【虚拟文件系统VFS】：①屏蔽了不同文件系统的差异和操作细节，为用户程序提供了文件系统操作的统一接口而无需关心底层文件系统的具体实现；②采用了**面向对象**的思想，定义了通用文件系统都支持的接口并要求下层的文件系统必须实现这些函数功能；③**只存在于内存中而不存在于任何外存空间中**，在系统启动时建立在系统关闭时消亡；④可以提高系统性能；

● 【文件系统挂载】：①初始化VFS中的挂载表、向VFS提供函数地址列表、将新文件系统加到挂载点（某个父目录）下；②UNIX使用系统的根文件系统（由内核在引导阶段直接安装），其他文件系统由初始化脚本或用户安装。

◇ UNIX本身是一个固定的目录树，只要安装就有，但是如果不给它分配存储空间，就不能对它进行操作，所以需要先给根目录分配空间才能操作这个目录树。

- 【操作系统引导】：①执行JMP指令跳转到$BIOS$；②开始执行$BIOS$的指令；③硬件自检并加载带有操作系统的硬盘；④在ROM中加载主引导记录MBR，MBR加载硬盘活动分区和分区引导记录PBR；⑤加载活动分区中的引导程序、加载启动管理器、加载操作系统内核。
- 【应用程序I/O接口】：①字符设备接口，以字符为单位、传输速率较低、不可寻址（只能使用顺序存取方式）、采用**中断驱动方式**；②块设备接口即**磁盘**，传输速率较高、可以寻址、采用DMA方式，通过内存的字节数组来访问磁盘而**不提供读/写磁盘操作**；③网络设备接口，使用socket系统调用创建**套接字**并通过此连接发送和接收数据、必须指明网络协议（UDP/TCP）；④阻塞或非阻塞I/O。
- 【阻塞或非阻塞I/O】：①阻塞I/O，执行I/O操作时会被阻塞、等待直到数据准备就绪才能进行读取或写入操作、**大多数操作系统提供的I/O接口都是采用阻塞I/O**；②非阻塞I/O，执行I/O操作时不会被阻塞（而是立即返回，无论数据是否准备就绪）、数据未准备好时应用程序可以继续执行并定期检查直到数据就绪；③异步I/O，发起I/O后可以继续执行其他任务无需等待、I/O操作完成后操作系统会通知程序并提供数据、需要使用特定的系统调用或库函数实现。
- 【设备分配方式】：①静态分配，主要用于对**独占设备**的分配、一次性分配、不会出现死锁但设备使用率低；②动态分配，在进程执行过程中根据执行需要进行、有利于提高设备利用率但有可能造成进程死锁。
- 【设备分配过程】：①根据物理设备名查找**系统设备表**SDT；②根据**系统设备表**SDT找到**设备控制表**DCT；③根据**设备控制表**DCT找到**控制器控制表**$COCT$；④根据**控制器控制表**$COCT$找到**通道控制表**$CHCT$；只有**设备、控制器、通道**三者都分配成功时这次设备分配才算成功，之后便可以启动数据传送。
  - ◇ 系统设备表→设备控制表→控制器控制表→通道控制表。
- 【缓冲寄存器相关计算题】：例，若在I/O接口的通信速率为9600bps设置的数据缓冲寄存器的宽度为8位，则I/O接口每【$\frac{8\text{bit}}{9600\text{bps}}=0.8\text{ms}$】就要中断一次CPU，且要求CPU在【$0.8ms\times\frac{1\text{bit}}{8\text{bit}}=0.1\text{ms}$】内响应。
  - ◇ 【$\frac{8\text{bit}}{9600\text{bps}}=0.8\text{ms}$】：思路同I/O中断方式的CPU中断时间；
  - ◇ 【$0.8ms\times\frac{1\text{bit}}{8\text{bit}}=0.1\text{ms}$】：必须在数据缓冲寄存器填满后**下一个比特数据到达前把数据移走**。

- 系统开机后操作系统被加载到内存中的系统区，这段区域是RAM。
- **软中断指令=自陷指令**。
- 通用寄存器清零不是特权指令、不需要切换到内核态执行。

- 时间片轮转实现了对于 CPU 的虚拟化，内存分区实现了对内存资源的虚拟化。
- 处理机被抢占会从运行态→就绪态，不会阻塞。
- 进程P由阻塞态转化为就绪态（唤醒）是由wakeup原语实现的，该原语是由进程P相关的进程（“**协作进程**”）调用的。
- 切换进程后 TLB 作废，Cache 一般不清空。
- 高低调度调作业，低级调度调进程。中级调度恢复挂起。
- **进程之间的切换是通过中断实现的**，有了中断才有了多任务处理系统。
- 【进程的撤销】：①正常结束后回收，是最主要的因素；②异常终止，包括越界、I/O故障、算术错误等；③外界干预，用户、操作系统、父进程等均可终止。
- 说法『~~**一个进程的状态发生改变时一定会导致其他进程的状态发生变化**~~』**是错误的**，当系统队列中只有其一个进程时。
- **【逻辑地址和物理地址】**：①堆栈指针指向的都是逻辑地址，**所有指针指向的可以认为都是逻辑地址**；②存储向量的寄存器（如页基址寄存器）均存储的是物理地址，避免循环依赖等问题（指向的地址又映射到自己）。
  ◇ 即可能存在循环依赖问题的是物理地址（即在内存中有物理存储的），否则就为逻辑地址。
- 交换区 swap：物理内存不足时，用于临时存储当前系统中不活跃的**页面或进程**。
- 进程优先级与分配资源和作业长短无关。
- 【进程控制块PCB】：①包含进程PID、进程状态、源程序程序段、数据地址、CPU现场保护区、堆栈指针、PC指针、FCB指针；
- **【线程】**：①最小的**分配和调度**单位，不拥有系统资源；②可以访问隶属于进程的资源，**共享**进程的地址空间，**没有独立地址空间**；③**同一进程中的线程切换不会引起进程切换**，不同进程中的线程切换则会引起进程切换；④只有内核级线程才有TCB。
- 子程序调用：一定会保存PC信息，而一定不会保存PSW的信息。
- **可重入代码/纯代码一定不是临界资源**。
- **【※程序载入过程】：编译、链接、装入(装载)、（执行）**。
  ◇ 其中，编译将程序翻译成汇编代码，链接将程序与库函数合并形成可执行文件期间地址变换机构进行地址重定位**将逻辑地址翻译成物理地址**。
- 【重定位】：静态重定位/**可重定位装入**在**装入时**进行地址转换（重定位），动态重定位在**执行时**进行地址转换（重定位）。
- 【同时分配内存空间、打印机和处理机的题】：①每个进程到达时即被分配可用资源；②**一个进程必须获得除 CPU 外所有所需资源才能参与进程调度**；③太复杂了一定要全写出来。

● 【伙伴算法】：①每次回收可能进行多次合并、每次合并只将两个相同大小的块进行合并；②回收一个大小为2^k的块时若已经存在一个同样为2^k的空闲块，则将其合并为一个大小为2^{k+1}的块并继续检查；

● 【内部碎片】：单一连续分配、固定分区分配、请求分页、段页式存储。

● 【外部碎片】：动态分区分配、请求分段。

● 外部碎片可通过紧凑技术使用动态重定位寄存器解决

● 【※没有**全局置换固定分配**策略】：因为**全局置换意味着进程拥有的物理块数量必然会改变**，因此不可能是固定分配。

● 【工作集】：对于某进程访问页面的序列，某时刻的工作集为该时刻之前 n 个访问的页面（去重），n 为工作集大小。

● 【改进型时钟置换算法】：在其他条件都相同时，应优先淘汰**有修改过的页面**，避免I/O操作。

◇ 没访问没修改，没访问**有修改**，有访问没修改，有访问有修改。

● open 系统调用：文件存放路径、文件名、要对文件的操作类型。

● read 系统调用：文件描述符、缓冲区首址、传输字节数。**不需要文件名和存放路径**。

● 对访问的文件执行 close 系统调用时，若引用计数值仍大于 0，即仍然存在进程使用该文件，则不能将文件写回外存。

● 【※※※以scanf( )为例，使用键盘输入数据时发生了什么】：①执行函数时触发系统调用、CPU切换至内核态后执行系统调用；②系统调用初始化后启动外设并**阻塞当前用户进程**直到键盘输入数据；③键盘中断例程（设备驱动程序）将数据从键盘控制器（I/O**接口**）传送到**内核缓冲区**后，**唤醒当前用户进程**；④最后系统调用返回，当前用户进程再次运行时将内核缓冲区的数据送至**用户缓冲区**。

◇ 中断例程结束后数据在内核缓冲区，系统调用返回后可以认为送往用户缓冲区。

● 【程序在运行中调用I/O，问进程状态变化】：应该完整回答执行I/O →阻塞→就绪→重新运行的过程。

● 对于任意文件系统，若用户类别有n种，访问权限有m中，则需要描述文件权限的位数至少为nm位。

● 多级目录结构中，从根目录到任何文件有且只有一条路径，与用户名和用户目录无关。

● 引入索引节点前目录项存放文件的FCB，引入后仅需要存储文件名和索引节点编号。

● 文件索引节点维护的是文件的物理结构。

● 【提高文件访问速度】：①提前读；②延迟写；③分配连续簇；④使用磁盘高速缓存；⑤内存映射文件。

◇ 提前读：使用**顺序访问**时，提前将下一块数据读入内存，故**对随机访问无效**。

◇ 延迟写：在内存中设置**页缓冲区队列**（属于共享资源），若进程再次访问其中的数据直接读出避免启动I/O。

- 文件索引节点是实现文件共享的其中一种方式、即**硬链接**。
- 【不同进程共享文件】：①可使用读或写的方法打开，操作系统不保证其互斥性；②系统打开文件表FTB只有一个对应的表项；
- 【不同进程共享表或段】：①每个进程各自的页表或段表指向共享内容的起始位置，在物理内存中只存在一份内容，但对应的段号或页号各自维护；
- 【**簇**】：①对**文件存储空间的分配以簇为单位**，所以**一个文件占用的空间一定是簇大小的整数倍**，文件小于一簇也要占用一簇的空间；②可以减少FAT表表项、降低FAT表占用空间、减少FAT表存取开销。
- 逻辑记录是对文件进行存取的基本单位。
- 【存储区前后均有足够的空间】：大概率前移。
- 每次要读或写一大批记录时，顺序文件的效率是所有逻辑文件中最高的。
- 【传送$ASCII$的设备（键盘、FTP）】：每传送传输一个$ASCII$字符，除了需要传输$ASCII$字符的七位外，还需要额外传输起始位、停止位、校验位各一位。
- 无论硬链接还是软链接都会导致每个共享文件存在重复的文件名，且每增加一个硬链接\软链接均会多增加一个文件名，故若**将目录中的所有文件全部转储到磁带等顺序存储载体上时会产生多个文件副本**。
- 每移动一个磁盘块需要访问磁盘两次（读+写）。
- 【最大文件长度】若地址或索引位数为m位，每块大小为N字节：①链接分配：【$2^m \cdot \left(N-\frac{1}{8}m\right)$】字节，尤其注意要减索引占用；②索引分配：【$\sum_{k=1}^{n}\left(\frac{N\times 8}{m}\right)^k \cdot N$】字节，其中$n$为间接地址项次数；
- 文件控制块需要额外存储的字段：链接分配【文件首地址、最后一块物理块块号】，连续分配【文件首地址、文件大小】。
- FCB 中有 A 个直接地址项和一、二、三次间址项各一个，若每页可以存储地址项 k 个，当前访问的逻辑页号为 n，则

◇ $n < A$：直接地址项；

◇ $A \le n < A+k$：一次间址项中第$n-A$个地址项；

◇ $A+k \le n < A+k+k^2$：二次间址项中一级索引为$\left\lfloor\frac{n-a-k}{k}\right\rfloor$，其中第$(n-a-k) \bmod k$个地址项；

◇ $A+k+k^2 \le n < A+k+k^2+k^3$：三次间址项中二级索引为$\left\lfloor \frac{n-a-k-k^2}{k^2} \right\rfloor$，一级索引为$\left\lfloor \frac{(n-a-k-k^2) \bmod k^2}{k} \right\rfloor$，其中第$(n-a-k-k^2) \bmod k$个地址项。

● 计算页表项或页目录项时对于类似【$\mathrm{LA}+0041\mathrm{H}\times 4$】的计算式，一般使用移位进行运算不需要将后半部分算出具体值。

● 【缓冲区平均耗时】：数据→缓冲区→用户区→处理的三个箭头耗时记为A、B、C，则单缓冲区为$\max(A,C)+B$，双缓冲区$\max[A,(B+C)]$。

● 统一编址：把I/O端口当做存储器的单元进行地址分配，用统一的访存指令访问I/O端口，靠不同的**地址码**区分内存I/O设备。

◇ 端口需要**占用主存地址空间**。

● 独立编址：使用专门的I/O指令访问I/O端口，靠不同的**指令**区分，速度快，不占用主存地址空间。

● $SPOOLing$必要条件：①**外存/硬盘**作为输入井和输出井；②多道程序**并发**处理技术；③$SPOOLing$软件；④进行虚拟独占设备。

● $SPOOLing$的输出井和设备之间存在一个内存中的输出缓冲区作为中介，没有直连数据通路。

● $SPOOLing$技术所用的输入进程和输出进程均是**和用户进程一起参与 CPU 调度**，均**运行在内核态下**，且其输出进程至少需要两种设备驱动程序的支持。

● 磁盘初始化

1. 低级格式化/物理格式化：**扇区的划分**。低级格式化为每个扇区**使用特殊的数据结构填充磁盘**（确定扇区校验码位数）。
2. 逻辑格式化：**创建文件系统，建立文件系统根目录**，初始化存储空闲磁盘块信息的数据结构（**空闲磁盘表**）。

● 【外存空闲磁盘管理】：空闲表法、空闲链接法、位示图法、成组链接法，还有索引节点也可表示空闲磁盘。

● 引导扇区在安装操作系统时写入，由操作系统决定。

● 寻道距离指的是磁道，若给出的是盘块号，则应当转化为对应磁道号

● 磁盘操作时间：总平均存取时间T_a=寻道时间T_s+延迟时间T_r+传输时间T_t。其中寻道时间T_s与磁盘调度算法密切相关。

1. **寻找时间/寻道时间**T_s：移动磁头，$T_s=s+m\times n$。
   ▲ 启动磁头臂是需要时间的，假设耗时为s。
   ▲ 移动磁头也是需要时间的，假设磁头匀速移动，每跨越一个磁道耗时为m，总共需要跨越n条磁道。

2. 旋转延迟时间T_r：通过旋转磁盘，使磁头定位到目标扇区所需要的时间。$T_r = \frac{1}{2} \times \frac{1}{r} = \frac{1}{2r}$。

▲ 磁盘转速为r，$\frac{1}{r}$就是转一圈需要的时间，找到目标扇区平均需要转半圈，因此再乘$\frac{1}{2}$。

3. 传输时间T_t：从磁盘读出或向磁盘写入数据所经历的时间。$T_t = \frac{b}{rN}$。

1. 其中磁盘转速为r，读/写的字节数为b，每个磁道上的字节数为N。

2. 每个磁道要可存N字节的数据，因此b字节的数据需要$\frac{b}{N}$个磁道才能存储，而读/写一个磁道所需的时间刚好又是转一圈所需要的时间$\frac{1}{r}$，$T_t = \frac{1}{r} \times \frac{b}{N}$。

● 周转时间：周转时间等于**完成时刻减到达时刻**，带权带权时间等于**周转时间除以运行时间**，平均周转时间等于**各个进程带权带权时间的平均值**。

● 磁盘调度算法

◇ 最短寻找时间优先算法$SSTF$可能导致饥饿。

◇ 扫描算法$SCAN$：回头时**沿途处理请求**。不会导致饥饿。

◇ **循环**扫描算法$C-SCAN$：回头时沿途**直接回到最前面**，不沿途处理请求。不会导致饥饿。

◇ 仅 FCFS 算法不会导致磁头黏着。

计算机网络

- OSI参考模型
  - ◇ **物理层**：解决使用**何种比特**来解决传输的问题（如何选物流车）、传输单位是比特。
  - ◇ **数据链路层**：解决分层在**一个网络或一段链路上**传输的问题（如何运输）、范围是相邻两个节点、传输单位是帧。
  - ◇ **网络层**：解决分组在多个网络上**传输/路由**的问题（如何分拣）、传输单位是IP数据报或分组、范围是整个网络或两个主机。
  - ◇ **传输层**：解决进程之间（**端到端**）基于网络的通信问题（如何派件）、传输单位是报文或报文段、范围是两个用户进程之间。
  - ◇ **会话层**：负责管理主机间的**会话**进程，包括建立、管理及终止进程间的会话。
  - ◇ **表示层**：主要处理在两个通信系统中**交换信息的表示方式**。
  - ◇ **应用层**：解决通过应用进程的交互来实现特定网络应用的问题（如何售后）。
- 各层的服务访问点SAP：数据链路层【以太网帧的**类型**字段】、网络层【IP 数据报的**协议**字段】、传输层【端口】、应用层【用户界面】。
- OSI模型中，下层为上一层提供服务。
- **传输媒体不属于物理层，传输媒体和物理层放一起就是错的**。
- TCP/IP参考模型：先有协议栈才有参考模型、常考**传输层**看清楚是哪个模型。
  - ◇ 网络接口层：作用是从主机或结点接收IP分组，并把它们发送到指定的物理网络上。对应OSI参考模型中的物理层和数据链路层。
  - ◇ 网际层：将分组发往任何网络，并为之独立地选择合适的路由，但它不保证各个分组有序地到达，各个分组的有序交付由高层负责。网际层定义了标准的分组格式和协议，即IP和IP协议。支持主机到主机的通信。
  - ◇ 传输层：和OSI参考模型中的传输层类似，即使得发送端和目的端主机上的对等实体进行会话。支持应用到应用的通信或进程到进程的通信。
  - ◇ 应用层：对应OSI参考模型中的会话层、 表示层和应用层。
- 两个模型的差异
  - ◇ OSI在网络层支持无连接和面向连接的通信，但在传输层仅有面向连接的通信（不是 TCP，定模型的时候也没有 UDP）；
  - ◇ TCP/IP模型认为可靠性是端到端的问题，因此它在网际层仅有一种无连接的通信模式， 但传输层支持无连接和面向连接两种模式。

● 【奈式准则】：①$\begin{cases} B=2W \\ R=2W\log_2 V \\ R=B\times\log_2 V \end{cases}$，其中$B$表示理想低通信道下的极限波特率、$R$表示理想低通信道下的极限比特率，奈式准则限定的是码元的传输速率B；②奈式准则意义是，在任何信道中，波特率是有上限的，超过该上限就会出现码间串扰（**悲观**）；③应对方法是使每个码元能携带更多的个比特量的信息。

● 【香农定理】：①$R=Wlog_2(1+S/N)$，其中S/N表示$\frac{S}{N}$纯数值无单位、可使用$\text{SNR}=10\cdot\log_{10}\frac{S}{N}$转化；②香农定理的意义在于，只要**信息传输速率/比特率**低于信道的极限传输速率，就**一定能**找到某种方法来实现无差错的传输（**乐观**）；③信噪比越大（信号越强）、**信息传输速率越小（传输每个码元的时间越长）**，数据的抗噪能力越强，判决错误的概率越小；④**有噪声的信道只能是香农定理**。

● 【CSMA/CD协议】：①边发送数据边**检测碰撞**，适用于半双工网络；②在2RTT时间内没有检测到碰撞，那么就可以肯定本次发送没有碰撞；③截断二进制指数规避算法：$n\times 2\tau$，其中n为0到2^{k-1}，k不超过10，最多16次超过就丢弃；④以太网规定争用期为51.2μs，最短帧长为$64\text{B}=512\text{bit}$，凡是小于则被判定为无效帧。

● 【停止等待协议SR】：①相当于发送窗口和接收窗口大小均为 1 的滑动窗口协议，只需要一位帧编号，**数据帧与确认帧必须编号**；②发送一个帧直到对方确认前均不会再发送；③发送方和接收方均设置一个帧缓冲区用于必须保存副本以便重传。

● 【后退N帧协议GBN】：①**仅GBN是累计确认**；②同时采用捎带确认，**确认报文在下一个数据帧上**；③由于接收窗口仅为一，故接收方只**按顺序**接受帧，除此以外的帧会被丢弃；④重传时必须**把原来已经正确传送的数据帧再次传输一次**；⑤信道利用率较高，但当信道的传输质量很差时会导致误码率较大，不一定优于SW。

● 【选择重传协议SR】：①由于接收窗口大于一，故不管是否按序都能接受传过来的帧（**不采用累计确认**，来一个确认一个），且仅SR的接收方有缓存；②只会重传出错帧。

● 【集线器Hub】：①网络**拓扑结构上属于星型，逻辑上属于总线型**；②相当于多端口的**中继器**；③只能在**半双工状态**下工作；④以太网上的集线器默认是100BaseT，即传输速率为100Mbps；⑤无法分割广播域和冲突域、不能连接两个具有不同速率的局域网。

◇ 中继器/转发器的实现原理是信号再生，而非简单地将衰减的信号放大；

◇ 采用粗同轴电缆的10BASE5以太网$5-4-3$规则：5是指不能超过五个**网段**、4是指在这些网段中的物理层网络设备（中继器，集线器）最多不超过四个、3是指这些网段中最多只有三个网段挂有计算机；

● 【以太网交换机】：①相当于多端口的网桥，可互联不同物理层、不同MAC子层和不同速率的以太网，且仅适用于少数据量的局域网，否则会**出现广播风暴**；②总带宽是各端口带宽之和即总容量会变为原来的N倍；③一般工作在**全双工方式**；④可以隔离碰撞域/冲突域，**不能隔离广播域**、可以连接两个具有不同速率的局域网（**需要存储转发**）。

◇ 若单接口速率为a，半双工情况下带宽为a，总容量为$\frac{1}{2}aN$；全双工情况下带宽为$2a$，总容量为aN。

● 【路由器】：①可以**抑制**网络上的广播风暴；②由路由选择和分组转发两部分构成，其中交换结构可以通过通过**存储器、总线和互联网络**进行交换，**不进行差错检测**；③路由器是面向协议的，而低级设备（网桥、交换机等）都与高层协议无关；④一般默认网关地址就是**离主机最近路由器**的LAN端口地址；

◇ 【路由器无法处理上层数据】：路由器要求物理层、数据链路层、网络层协议可以不同，但网络层以上的高层协议必须相同。

◇ ※※**路由器每个接口都需要分别配置IP地址**，尤其在IP分配题中需要特别注意；

◇ 【特定路由】：互联网默认路由0.0.0.0/0、特定目的主机专门制定的路由指定子网掩码为32；

● VLAN等价于广播域，网段等价于冲突域。

● 【IP分组的转发过程】：①检查校验和字段，出错直接丢弃（不会发送差错报文）；②根据目的IP地址查找路由表，采用**最长前缀匹配法**确定下一跳，找不到直接丢弃并发送ICMP目的不可达差错报文；③更新IP首部，更新TTL和检验和，如果是NAT路由器还需要更新IP地址等。

◇ 差错报文主要用于报告如目的不可达、超时等**网络层逻辑问题**，而不是物理层或链路层的传输问题（如校验和错误）。

● 【分组与排序】：IP数据报**在网络层进行分片和重组**、在**传输层中**依靠序号字段**进行排序工作**。

● 【ARP协议的通信过程】：①检查ARP高速缓存，如果有对应项就直接写入MAC帧并结束；②用FF-FF-FF-FF-FF-FF 作为目的MAC地址，并广播ARP**请求分组**；③目的主机收到请求后会向源主机**单播ARP响应分组**，源主机收到后将该映射写入ARP缓存中。

◇ ARP请求报文直接封装在以太网广播帧中、使用广播 MAC地址全一；ARP**响应分组封装在以太网单播帧中**、 MAC**地址不为全一**。

◇ ARP工作在网络层（能看到IP地址）、NAT工作在传输层（能获取端口）。

● 【子网划分】：可用子网数量不用减二、可分配主机数量应该减二。

● RIP是应用层协议，使用UDP；$OSPF$是网络层协议，直接用IP数据报传送；BGP是应用层协议，使用TCP。

- 【路由信息协议RIP】：基于**距离-向量路由算法**、一条路径最多只能包含15个路由器（16表示网络不可达）、慢收敛是导致发生**路由回路**的根本原因、一个RIP报文最多只包含25个路由。
- 【解决慢收敛方法】：①抑制计时器法；②毒性逆转，将度量值变为无穷大的数，然后将其中毒的路由信息发布出去；③触发刷新，检测到网络故障的路由器会直接发送一个更新信息给邻居路由器，并依次产生触发更新通知它们的邻居路由器（**能使全部路由器在最短的时间内收到更新信息**）；④分割水平线；⑤限制路径最大距离。
- 【开放最短路径优先协议OSPF】：使用**洪泛法**向所有相邻的路由器发送信息（不再发送给刚刚发来信息的那个路由器）、**链路状态报文大小与网络中的路由结点数目无关**（链路状态算法比距离-向量算法有更好的规模可伸展性）。
- 【边界网关协议BGP报文】：使用KEEPALIVE作为OPEN的确认。
- **本机的DNS地址即为本地域名服务器**。
- 【HTTP】：访问的 Web 网页附有 k 个小文件，小文件大小均为一个MSL，则
  - ◇ 首先先看清楚是从点击开始、从发送请求开始、从连接建立开始；
  - ◇ 三种情况共有：建立TCp链接时前两次握手正常，第三次主机向服务器发送确认握手时会附带第一次对于 Web 网页的请求，服务器会在稍后发送确认时捎带 Web 网页，总高耗时 2RTT。
  - ◇ **非持续非流水线、HTTP/1.0**：每传输一个 Web 页面或者小文件都需要重新建立一次TCP链接，第三次握手可以附带HTTP请求。**每一个 Web 页面和小文件都需要至少两个 RTT**。
  - ◇ **持久连接非流水线、HTTP/1.1**：客户在收到前一个响应后才能发出下一个请求；
  - ◇ **※持久连接流水线、HTTP/1.1 默认**：默认TCP链接的第三个报文携带HTTP请求报文，之后发送的数据长度依照慢开始（假设理想条件没有重传和拥塞）算法进行传输。
- 【**※※※并行TCP连接**】：几个小图像就是几个并行的TCP连接。即若采用并行TCP连接，使用非持续非流水线链接传输时，只需要4RTT其中前两个RTT用于前两次握手，第三次链接使用并行TCP连接一次性全部传输完成。
- 【HTTP】：①实体主体部分可以为空，例如404和HEAD请求；②一个请求报文对应一个响应报文；③一个TCP报文只能封装一个HTTP请求报文。
- 【点对点协议PPP】：①**不可靠，提供差错检测但不提供纠错功能、不使用序号和确认机制**；②仅支持点对点的链路通信，不支持多点线路；③只支持全双工链路；④两端可以运行不同的网络层协议；⑤面向字节；⑥**异步传输时使用字节填充法，同步传输链路时采用零比特填充法**。
- 【PPP帧】：无须采用$CSMA/CD$协议（不是总线形、同时没有最短帧），有效信息最小值为0B而非通常的46B。

● 【高速以太网（快速以太网）】：速率大于100Mbps的以太网。**半双工时使用$CSMA/CD$协议**、采用保持最短帧长不变而将最大电缆长度减少到$100m$的方法。

◇ 【100BaseT以太网】：使用双绞线、星型拓扑、全双工方式工作下无冲突；

◇ 【吉比特以太网】：使用光纤或双绞线；

◇ 【10吉比特以太网】：使用光纤、只支持全双工故不使用$CSMA/CD$协议。

● 【移动IP】：固定主机B向移动主机A发送IP数据报，则该数据报的源IP地址为主机B的IP地址，目的IP地址为移动主机A的永久IP地址，归属代理收到该IP数据报后，会将目的IP地址改变为外地代理的转交地址。

● 【DHCP】：①有发现报文、提供报文、请求报文、确认报文四种报文类型②发现报文和请求报文目的MAC地址均为广播地址全一，源IP地址为全零；③DHCP**发现报文**封装在广播IP数据报中，使用广播IP地址全一， 之后封装在以太网广播帧中，使用广播 MAC地址全一；④DHCP相关报文的目的地址均为广播地址（全一），其仅DHCP是这样。

◇ 在DHCP执行初期，客户端不知道服务器端的IP地址，而在执行中间，客户端并未被分配IP地址。

● 【IEEE802.11无线局域网】：①移动站A如果要和另一个基本服务集中的移动站B通信，就必须经过两个接入点AP_1和AP_2，即$A \to AP_1 \to AP_2 \to B$，其中$AP_1 \to AP_2$的通信是使用有线传输的；②**静态路由协议、RIP、OSPF、BGP均不适用于移动自组网络**；③※※※IEEE802.11的MAC帧格式，**地址一为接受地址、地址二为发送地址**；④发送过程中不需要进行冲突检测，故不使用CSMA/CD协议，而使用CSMA/CA协议。

● 【虚拟局域网VLAN】：①是一个**广播域！广播域！广播域！**；②部署在**数据链路层**、通过标识符实现逻辑不需要额外的硬件支持；③可通过交换机端口、网卡MAC地址、网络层IP子网地址、协议、策略进行划分。

● 【IP多播（组播）】：组播时延远小于多份单播、仅**使用UDP不可靠**、组播地址（D类地址）只能用于目的地址。

● 【网际组管理协议IGMP】：①网络层协议；②管理多播组成员关系，确保多播流量只发送到有需求的网络部分；③组成员关系是动态的，需要周期性探询本地局域网上的主机以便确认是否还是多播组成员。

◇ 不能知道IP组播组包含的成员数，也不知道这些成员分布在哪些网络上；

◇ 只能知道本局域网上的组播组成员，并不能在因特网范围内对所有组播组成员进行管理。

● 【TCP的四种计时器】

1. 重传计时器（超时计时器）：发送一个报文段时判断该报文段**是否需要超时重传**；

2. 持续计时器：当发送方收到一个**窗口大小为零的确认**时，为避免双方死锁等待，就启动持续计时器；

3. 保活计时器：使用保活计时器可**避免服务器在客户机突然出现故障时一直等待**；

4. 时间等待计时器：在连接释放时使用，为2MSL。

● 【BGP协议工作原理】：①自治系统的管理员选择至少一个路由器作为BGP发言人；②发言人使用TCP**链接**与其他自治系统中的发言人并交换Open报文以建立eBGP会话（使用Keepalive报文回复Open报文）；③使用Update报文交换路由信息（网络可达性信息）；④发现异常时，发送Notification报文终止会话。

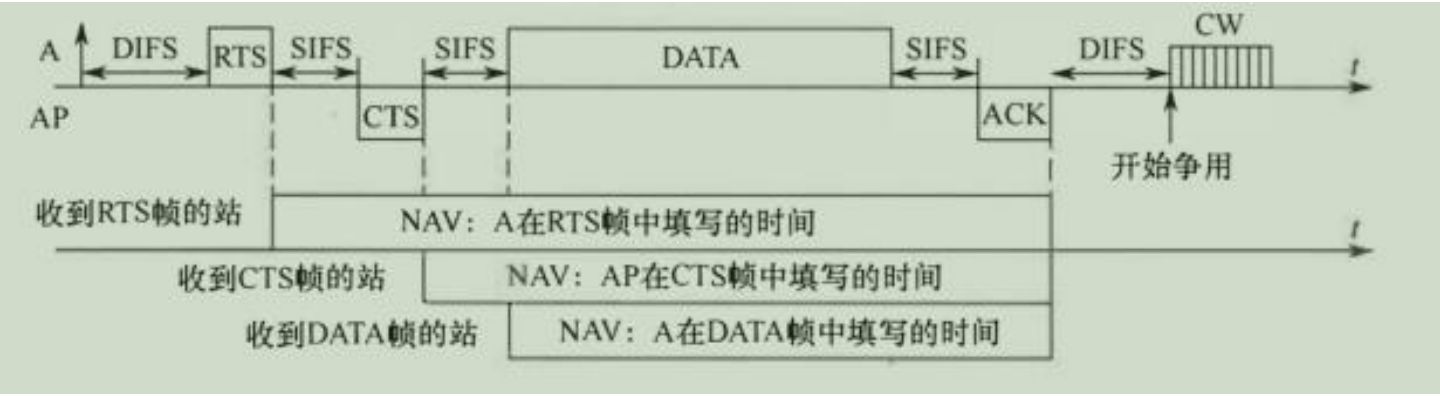
◇ eBGP会话用于不同自治系统之间，iBGP用于同一自治系统之内；

◇ Exterior外部的，Interior内部的。

● 【网络分配向量NAV】：在一个自治系统中，主机A向AP发送数据帧，B为自治系统下的一个隐藏栈，由于发送RTS帧和返回CTS帧均为广播，则需要在发送的帧中填写**该帧占用信道的时间**。

◇ A向向AP广播数据帧前会先广播RTS帧，将NAV = SIFS + CTS + SIFS + DATA + SIFS + ACK写入帧头；

◇ 当AP收到RTS帧时会广播一个CTS帧，将NAV = SIFS + DATA + SIFS + ACK写入帧头；



● 【使用十六进制列出IP数据报全部内容的大题】：其中A、B为题中考查的通信双方，且A为题上展示IP数据报的那方。

1. 分析清除每个报文的通信双方，**可能有捣乱的**：①A发送的报文一般对方都是B，且A、B的协议字段都应该是TCP；

2. SYN仅在TCP前两次握手才有效，第三次握手报文可以通过A第二次向B发送的报文判断；

3. 握手完毕后，若B向A发送数据，其首字节编号即为第三次握手时的确认号；

4. 应用层数据：①TCP数据报头之后的那一串数据均可视为应用层数据，直接算出其长度；②**四挥手第一次挥手**报文的序号减去**三握手第三次握手报文**的序号（需注意第三次握手是否发送了数据）；

- 特殊规定的数值
  - ◇ 以太网规定争用期为$51.2\mu s$，以太网最小帧长为64字节。
  - ◇ IP 数据报首部首部长度、总长度、片偏移，基本单位分别为 4B、1B、8B。
  - ◇ 【熟知端口号】：FTP数据连接20、FTP控制连接21、$SMTP=25$、$HTTP=80$、$POP3=110$。
- 【虚电路】：①只是逻辑上的电路，**不需要分配宽带**；②面向连接，可靠传输，**可确保有序到达**；③连接建立时进行路由选择，路由器记录虚电路号VCID与下一跳信息；④传输时依据虚电路号转发（即传**输路径固定**）⑤主要缺点是存在链接建立时间。
- 【数据报】：①无连接，不保证有序，不保证可靠，依靠上层协议实现可靠性，可能丢失或重复；②网络层使用IP协议；③每个报文的**传输路径**均依据路由选择**动态变化**；④到达后需要进行排序去重等。
- ※**总时延=发送时延+传播时延+排队时延+处理时延**。（以下内容暂不考虑排队时延）
  - ◇ 电路交换时延：①连接建立时延；②链路传播时延；③发送时延；
  - ◇ 分组交换时延：①源节点发送时延；②$n-1$个结点的转发时延（n表示链路数），每个结点的转发时延等于分组长度÷数据传输速率，即只分别计算一个分组的时延；③链路传播时延；
- 软件定义网络SDN
  - ◇ 控制平面：**路由选择**，贴近应用，所以在上面，即**对应北向接口**；
  - ◇ 数据平面：**分组转发**，贴近**交换机**，所以在下面，即**对应南向接口**。
- 【流量控制】

控制类型	作用范围	实现方式
数据链路层	点到点，两个相邻节点之间	收不下就不回复确认（确认控制帧）
网络层	作用于整个网络	使用**拥塞控制算法**检测拥塞并采取措施
传输层	端到端，作用于两台主机上相应的端口	接收端发送拥塞窗口

- 【**以太网传输介质**（以100BaseT为例）】：①速率，单位为Mbps，100表示100Mbps，若为10G即表示10Gbps；②Base固定，表示基带传输；③最后的字母表示介质类型，T双绞线$Twisted\ Pair$、F光纤$Fiber$、5粗同轴电缆（500 米长所以粗）、2细同轴电缆（200 米长所以细）。
- 数字数据调制为模拟信号：调幅ASK（振幅$Amplitude$），调频FSK（**频率**$Frequency$），调相PSK（**相位**$Phase$），正交振幅调制QAM。

- 静态划分信道不会发生碰撞。
- 静态划分信道：频分多路复用FDM，时分多路复用TDM，统计时分复用$STDM$，波分多路复用WDM，码分多路复用CDM。
- 码分多路复用CDM计算的内积若为零，则表示该站点没有发送数据。
- $CSMA$协议三种监听算法
  - ◇ 非坚持和$1-$坚持在空闲时均为直接发送，$p-$坚持和$1-$坚持在忙时均为持续监听。
  - ◇ 非坚持：信道忙时会**等待一个随机时间**之后再进行监听，**空闲立刻传输**；
  - ◇ $p-$坚持$CSMA$：道忙时会推迟到下一个时隙再监听，信道空闲时以p概率直接传输，概率$1-p$等待到下一个时间槽再侦听。
- $CSMA/CA$协议：①可以**避免碰撞**，且可以用于无线局域网即**无线传输**，这是与$CSMA/CD$（检测碰撞、有线传输）最大的区别；②使用交换RTS帧和CTS帧的方式预约信道。
- $CSMA/CA$协议帧类型：长度依次递增，优先度依次降低
  - ◇ $SIFS$（短IFS）：最短的IFS，用来**分隔对话帧**，使用$SIFS$的帧类型有ACK帧、CTS帧、分片后的数据帧，以及所有回答AP探询的帧等。
  - ◇ $PIFS$（点协调IFS）：中等长度的IFS，在PCF操作中使用。
  - ◇ $DIFS$（分布式协调IFS：最长的IFS，用于**发送数据前**。
- 截断二进制指数规避算法：$n\times 2\tau$，其中n为0到2^{k-1}，k不超过10，最多16次超过就丢弃。
- 以太网规定争用期为51.2μs，最短帧长为$64\text{B}=512\text{bit}$，凡是小于则被判定为无效帧。
- ICMP 报错的【源点抑制】报文：翻译一下就是让【发送方主机（源点）少发点（抑制）】。
- `ping` 使用 ICMP。
- IP 分组在经过路由器分片时会发生变化的字段：①标志字段 MF、DF；②片偏移；③总长度；④检验和。
- IP报文分片：①标识Identification，相当于组号、用于标记同属一组的IP报文；②标志Flag，中间位DF表示是否禁止分片、最低位MF表示后面是否还有分片；③片偏移，单位为$8B$；④总长度，单位$1B$、最大为$2^{16}-1B$但会受到MTU的限制；⑤首部长度，单位为$4B$、最小值为5（即IP报文最小值为$20B$）；⑥首部检验和、填充，分片时可能变化，填充会把IP报文对齐成四字节的整数倍。
- 【IP报文协议字段】：$\text{ICMP}=1$、$\text{TCP}=6$、$\text{UDP}=17$、$\text{OSPF}=89$。
- IP 多播转化为硬件多播时：前 25 位固定（`01-00-5E`），IP **后 23 位**映射到 MAC 地址上，范围为 `01-00-5E-00-00-00~01-00-5E-7F-FF-FF`，且注意IP表示为十进制、MAC地址为十六进制。
- 网络地址（主机号全为零）既不能作为源地址也不能作为目的地址，`127.xx.xx.xx` 既可以作为源地址也能作为目的地址。

● 一般默认 TTL 最大为 64，故两个使用 IP 通信的站点之间 TTL 不超过 63；RIP 最大为 16 跳，使用 RIP 的 TTL 不超过 15。

● LSI 为 OSPF 路由协议使用的**路径成本**（由链路的带宽计算得出），需要与 RIP 协议使用的路由器跳数（**距离**）相区分。

● 当 **TTL** 减为 0 时，路由器丢弃该报文并向源主机发送**发送时间超过的 ICMP 差错报文**。

◇ 所以若两个网络之间相隔最少 **n** 个路由器，数据报的 **TTL** 至少应为 **n + 1**。

● 【路由器找不到匹配的路由项】：丢弃该报文并向源主机发送**目的网络不可达的 ICMP 差错报文**。

● 【时延带宽积】：①表示传输某数据时链路上**最大比特数量**，等于传播时延×带宽（信道最大数据传输率），单位为 bit；②

● 若两主机通过 WIFI 连接，则其对应的路由器为AP。

● NAT 转换：在两个不同网络下有主机 AB，分别由不同的 **NAT** 路由器R_A, R_B管理。当主机 A 发送 IP 数据包到主机 B 时

1. 主机 A 发送数据包：源 IP 地址为主机 A 的私有 IP（**大概率需要分配**），目的 IP 地址为主机 B 的**公网 IP**。

   ▲ 若主机 B 为服务器，则该公网 IP 即为题上所给。

2. 数据包到达路由器R_A：将源 IP 地址转换为路由器R_A转发端口的公共 IP。
3. 传输过程中，路由器继续保持**源 IP 和目的 IP 不变**，但 **MAC 地址会随着数据包经过的路由器进行变化**。
4. 数据包到达路由器R_B：路由器会将目的 IP 地址转换为路由器R_B**内部网络**的 IP 地址，之后主机 B 收到数据包。

● IPv6 与 IPv4 的相比：①地址空间扩大，由32位变为128位；②不需要$DHCP$协议，支持自动配置；③**彻底移除校验和字段**，减少路由器处理时间；④首部长度固定$20B$，废除首部长度字段；⑤使用**拓展首部**取代可选字段；⑥**废除片偏移字段，只允许源主机分片**；⑦废除协议字段、废除总长度字段。

● 若大题中给出具体 xx 的表结构，则在题中【对 xx 进行配置】时一定会用到。

● 捎带确认：说明**确认帧帧长等于数据帧长**。

● 【分配IP地址】：①注意路由器端口也需要分配；②在没有要求必须要用完全部地址时，仅需保证每一组的IP地址没有重叠（即不存在一组IP是另一组的前缀即可）即可。

● 【网际控制报文协议ICMP】：作为IP报文的一部分进行传输。

- 【ICMP差错控制报文】：①终点不可达，路由器或主机**不能交付数据报**；②源点抑制，路由器或主机由于**网络拥塞**而丢弃数据报；③改变路由（重定向），由于**存在更好的路由**，路由器改变路由；④时间超过，生存时间TTL = 0；⑤分组过大，IPv6超过分片大小。
- 信道利用率：需要乘最大发送窗口，注意捎带确认（没说忽略确认帧就不要忽略）。
- 协议由语法、语义和同步三部分组成
  - ◇ 语法：语法规定了通信双方彼此“**如何讲**”，即规定了传输数据的格式；
  - ◇ 语义：语义规定了通信双方彼此“**讲什么**”，即规定了所要完成的功能，如需要发出何种控制信息、完成何种动作及**做出何种应答**；
  - ◇ 时序/同步：规定执行各种操作的条件、时序关系等，即事件实现**顺序的详细说明**，类似操作系统中的同步操作。
- 【$IEEE$802.11的MAC帧格式】：第一个地址为**接受地址**，第二个为**发送地址**。
- 静态介质访问控制不会造成冲突：频分多路复用FDM，时分多路复用TDM，波分多路复用WDM，码分多路复用CDM。
- UDP和TCP都有伪首部。
- 【使用UDP的协议】：DNS、RIP、TFTP、RIP、DHCP。
- 【使用TCP的协议】：HTTP、SMTP、FTP、SSH、BGP。
- 【直接使用IP的协议】：OSPF、ICMP、ping。
- UDP 首部：以**字节**为单位，最短 8B。使用目的端口号实现分用，检验和检测整个 UDP 数据报是否有错。
- 片偏移：最大数据长度×之前发送的报文数÷ $8B$。
- 【UDP检验和】：①将伪首部、UDP首部和数据部分分为16位一组，并将每一组按位相加（相加的过程中**如果最高位有溢出需将溢出位加回低位**，称为回卷）；②对最终获得的结果按位取反即得UDP检验和；③接收方接受时，重新计算以上过程但不取反，和传来的检验和相加为全一即通过。
- TCP面向字节流：TCP从上层接受数据的基本单位是数据块，向上层传递的基本单位是字节流，而 TCP 将其均视为字节流。
- TCP连接中连续通信中，接受方收到第 k 个报文段发回的确认报文中的确认号应与第 k+1 个报文段的序号相等。
- TCP滑动窗口自成一派，即使用的是可变大小的滑动窗口。
- TCP传输时，若在重传倒计时到期前接收到了更大确认号的报文，则不会重传。
- TCP数据报：①为$4B$整数倍，最短为$20B$。
  - ◇ 确认号：表示期望收到对方下一个报文段的第一个数据字节的序号；

◇ 数据偏移：以$4B$为单位。

- TCP三个标志位：①确认位 ACK，在连接建立后所有传送的报文段都必须把 ACK 置为1，**在除了三握手的第一次握手以外的整个 TCP 传输中恒有效**；②同步位 SYN，三握手时有效；③终止位 FIN，四挥手时有效。
- 【TCP三握手】：①双方均只消耗一个序号，**第三次握手可以不消耗序号**；②三握手全程一半不携带数据，但第三次握手也可以携带数据（例如 HTTP 的捎带请求），携带数据时需要消耗序号。
- 若TCP建立连接时，即三握手已经完成时，的序号为 seq=a，且在之后传输了k个MSS的段后，预发送的序号/对方的确认号即为$a + k \times MSS$。
- 【TCP四挥手中的状态】：①TIME_WAIT 状态双方共有，但只有主动结束方需要等待 2MSL；② FIN_WIAT_1 和 FIN_WIAT_1 均为释放连接发起方独有的状态。
- 【TCP四挥手最短用时】：发起方为H_{ost}，接受方为S_{erver}。考虑最短用时假设发起时数据已经全部传输完毕，则$\begin{cases} T_S = 1.5 \times RTT \\ T_H = RTT + 2MSL \end{cases}$。
- TCP拥塞控制中若接受方会对接受到的每一个段进行接受窗口通告，则可以认为**每接受一个段接受窗口就减一**，并非一次传输全部完毕后才减。

◇ TCP 中第八个段不是第八次传输。

- 对于TCP拥塞控制，发生快重传（接收到连续三个冗余确认帧）时拥塞窗口只降为当前一半，若超时则直接置 1，慢开始门限均降为当前一半。
- B/S模型中，**服务器端可在连接建立时主动发送信息**。
- 若从开机开始算起，某主机访问互联网前一定会使用DHCP获取公网IP。
- DHCP用于配置 IP 地址、子网掩码、DNS 等，有**发现报文、提供报文、请求报文、确认报文四种报文类型**。

◇ 其中**发现报文和请求报文的目的 IP 地址和目的 MAC 地址均为广播地址（全 1），源 IP 地址为全 0**。

- FTP服务器端口：数据连接 20，控制连接 21。
- 熟知端口：SMTP25，POP101，HTTP80，SMTP55。
- 若题上给出了域名，则最多访问次数为【.】个数加一（根域名服务器）。
- 权限域名服务器不是指需要权限，对其进行域名请求时若无法得出最终结果会继续往下查。
- 【广播】：①优点是维护简单成本低廉、服务器流量负载极低（服务器不需要向每个客户机单独发送数据）；②缺点是**广播禁止在**Internet **宽带网上传输**、客户端的最大带宽等于服务总带宽。

- 【组播】：①缺点是没有纠错机制；②优点是具备广播所具备的优点、服务端的服务总带宽不受客户机接入端带宽的限制、允许在Internet宽带网上传输。
- 【文件传送协议FTP】：①使用TCP、基于C/S模型；②控制连接（端口21）在整个会话期间一直保持打开状态，控制信息以七位$ASCII$格式传输；③数据连接只保持一段时间，数据传输完成之后就断开；④主动连接时服务器端数据连接端口号固定是20，**被动连接时服务器端端口号不确定**（由两者协商得到）；⑤可以传输$ASCII$和二进制。
- 【简单邮件传送SMTP】：端口25，使用TCP、C/S模型，只能传输7位$ASCII$码。
- 【邮局协议$POP3$】：端口110，使用TCP、C/S模型，使用明文在传输层上传输密码，不进行加密。
- 数据包的逐层封装
  - ◇ 【以HTTP为例】：HTTP报文、TCP报文、IP数据报、$CSMA/CD$报文；
  - ◇ 【以$DHCP$为例】：DHCP报文、UDP报文、IP数据报、$CSMA/CD$报文。

- 数据报方式和虚电路方式都属于分组交换、报文交换和分组交换都属于存储转发。
- 高速链路，提高的仅是**数据发送速率**而非比特在链路上的传播速率，仅减少数据的发送时延。
- 物理层的接口特性：主要记住均是关于接口、接线器、数据传输的，**不涉及具体的传输介质**。
- 会话层可以使用校验点实现数据同步、表示层可以对数据进行压缩、加密和解密。
- 数字信号的编码：
  - ◇ 仅非归零编码NRZ中间没有跳变，没有检错功能、无法传递时钟信号、需要带有时钟线；
  - ◇ 归零编码RZ，折一平零；反向非归零编码$NRZI$，折零平一，USB2.0**默认编码**；
  - ◇ 曼彻斯特编码，**中间跳变**，以太网默认编码方式（**以太网的波特率是数据率的两倍**）；差分曼彻斯特编码，码元之间跳变。
- 采样定理：模拟数据到数字信号时，采样的频率大于或等于模拟数据的频带带宽的**两倍**时，所得的离散信号可以无失真地代表被采样的模拟数据。
- 模拟信号调制：调幅ASK，**振幅**$Amplitude$；调频FSK，**频率**$Frequency$；调相PSK，**相位**$Phase$；正交振幅调制QAM，见下。
- 正交振幅调制QAM：将ASK与PSK结合起来。若波特率为B、采用m个相位、每个相位有n种振幅，则该QAM技术的数据传输速率R为$R = B\log_2(nm)$。
- 循环冗余码CRC：注意使用的是按位异或、FCS的生成与接收端CRC检验均由硬件完成；
- **静态划分信道不会发生碰撞**。

- 静态划分信道：频分多路复用FDM、时分多路复用TDM、统计时分复用$STDM$、波分多路复用WDM、码分多路复用CDM。
- 【令牌环网】：①逻辑上是环型的，但是物理上是星型的；②令牌不包含任何信息、使用TCP转发；③无视负载，轮流发送数据；④通过修改标志位并附加数据获取令牌并发送数据。
- 【局域网LAN】：①使用广播信道、可以广播与组播、双绞线为主流传输介质；②可以默认所有以太网都符合$IEEE$802.3系列标准规范，逻辑拓扑是总线型，物理拓扑是星型或拓展星型，使用$CSMA/CD$。
- 【以太网/802.3局域网】：①**广播**形式发送、使用**曼彻斯特编码**、网络中存在大量的广播信息导致性能降低；②**在使用静态MAC地址的系统中，如果有重复的硬件地址，那么这两个设备都不能正常通信**；③以太网适配器上有处理器和存储器，包括RAM和ROM，ROM上有MAC地址。
- 【以太网$V2$帧】：①在物理层插入前导码用于时钟同步，包括**前同步码**与**帧开始定界符**两个部分；②校验码采用32位循环冗余码CRC，但是不用校验MAC帧的前导码；③上层数据最短为46B。
- 【广域网】：比局域网大、在下三层均有分布（局域网仅有两层）、用于资源共享。
- 【虚拟专用网VPN】：是利用公用的互联网作为本地各专用网之间的通信载体、需要对数据进行加密。
- 【IP广播】：①受限（有限）广播地址，IP地址为全一，名义是整个IP网络的网络广播地址，但是由于路由器对广播域的隔离作用，实际表现为对本网络的广播；②直接广播地址，即网络地址将主机号位全部置一。
- 【不应发送ICMP差错报文的情况】：ICMP差错报文本身、第一个分片的数据报片之后的所有后续数据报片、组播地址、127.0.0.1、0.0.0.0。
- 【ping命令】：使用$ICMP$回送请求和回答报文测试连通性、工作在应用层但不使用TCP或UDP直接使用ICMP。