# DIVIDE-AND-CONCUER

Jiayuan Mao. maojiayuan@gmail.com

ITCS, Institute for Interdiscinplinary Information Sciences, Tsinghua University.

# What is?

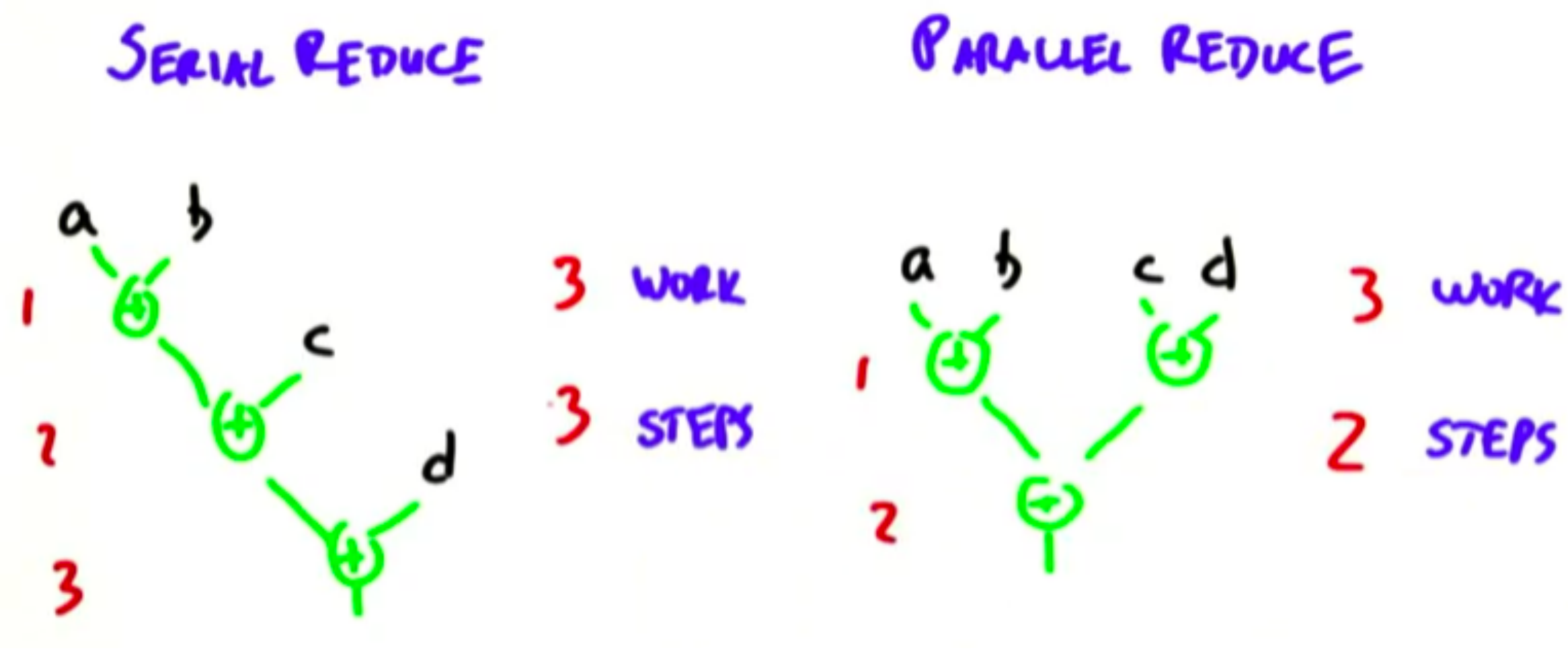
- A divide and conquer algorithm works by :
  - recursively breaking down a problem into
  - two or more sub-problems of the same or related type
  - until these become simple enough to be solved directly.
- Paradigm:
  - Divide
  - Conquer
  - Reduce

# Reduce Algorithms: Non-Trivial!

- Strongly connected with concurrency algorithms.
- E.g., in industrial standard: Map-Reduce.
  - We want to compute: 1^2+2^2+3^2+4^2+5^2.
  - Map: [1, 2, 3, 4, 5] -> [1, 4, 9, 16, 25]
  - Reduce: [1, 4, 9, 16, 25] -> 55
- Give a set of intermediate results, how to obtain the final result?

# For FUN. Concurrency Reduction Algorithms: Selected.

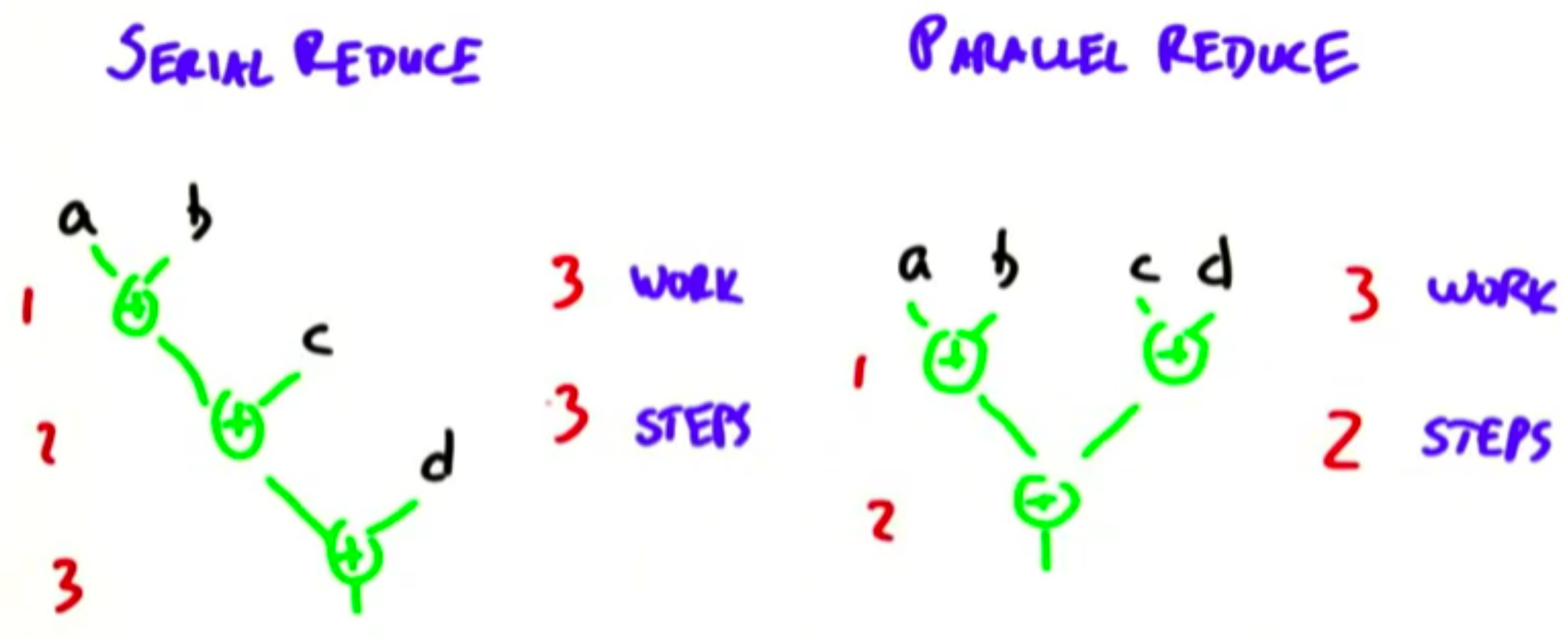
- Reduce-Sum
- Design rules:
  - “Parallel”.
  - “Use all CPU cores”.




Figures from: http://blog.csdn.net/abcjennifer/article/details/43528407

## For FUN. Concurrency Reduction Algorithms: Selected.

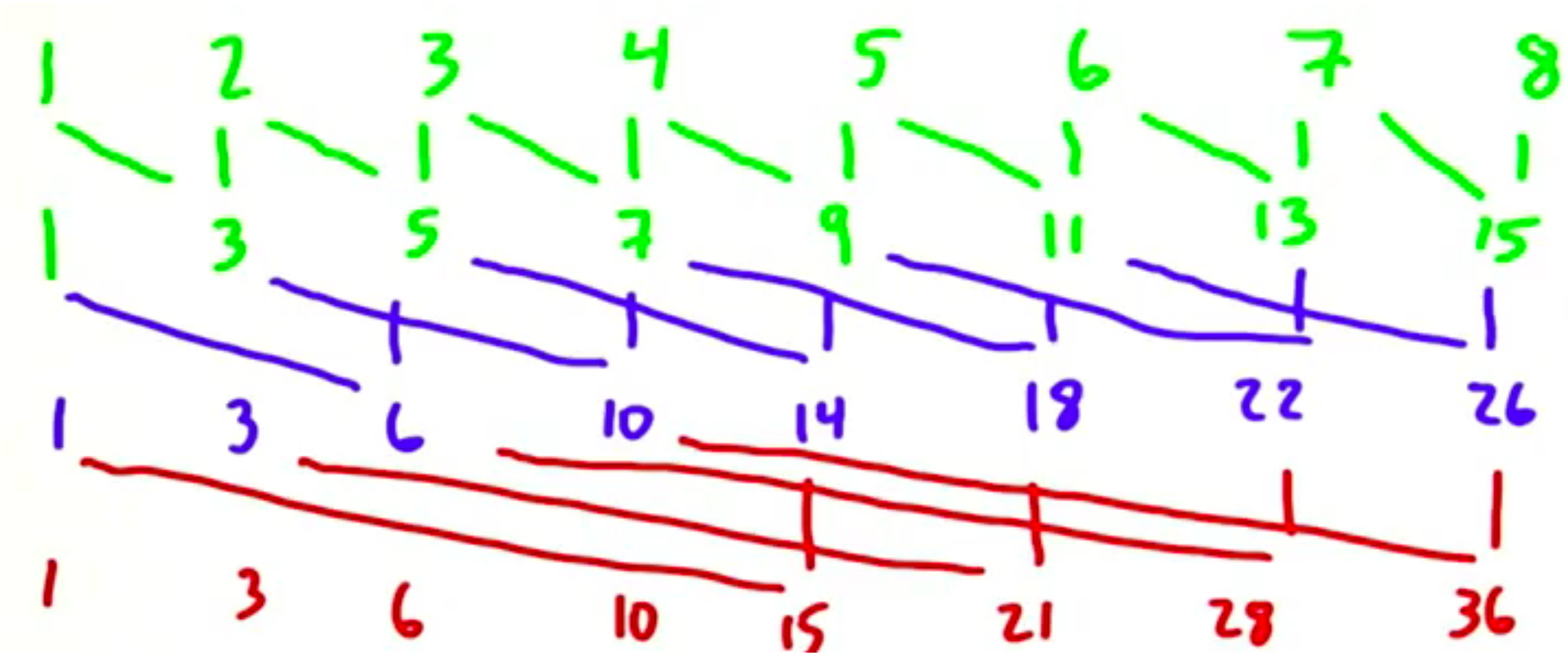
- Reduce-Sum




Figures from: http://blog.csdn.net/abcjennifer/article/details/43528407

# For FUN. Concurrency Reduction Algorithms: Selected.

- Reduce-Cumsum




- Hillis-Steele Algorithm
- O(n log n)

Figures from: http://blog.csdn.net/abcjennifer/article/details/43528407

# Concurrency Algorithms: More Efficient Sorting.

- An OK Concurrency Sorting: MergeSort.
- Why MergeSort is not good?

# Concurrency Algorithms: More Efficient Sorting.

- An OK Concurrency Sorting: MergeSort.
- A better solution: Bitonic Sort. 双调排序

- 双调序列：先单调递增，再单调递减的序列。（或反过来）
- 双调序列的排序：左右分成两段，同时从左到右枚举，小的放在左边。就可以得到两个双调序列。
- 并且：左边的双调序列中元素全部小于右侧得到的双调序列的所有元素。
- 迭代这个过程，长度为2的双调序列=有序。

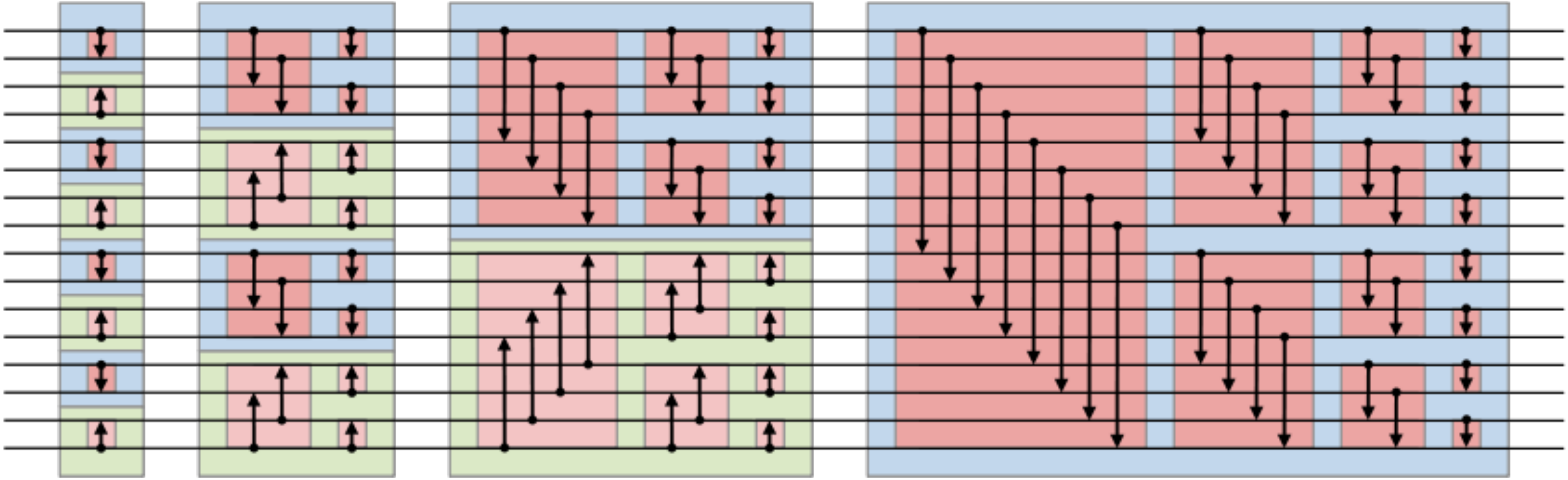
# Concurrency Algorithms: More Efficient Sorting.

- An OK Concurrency Sorting: MergeSort.
- A better solution: Bitonic Sort. 双调排序

- 如何得到一个双调序列？同样Divide-and-Conquer.
- 相邻元素组成单调序列。（相邻单调序列排序方向相反）
- While True:
  - 相邻单调序列（通过排序）变成新的单调序列。

# Concurrency Algorithms: More Efficient Sorting.

- An OK Concurrency Sorting: MergeSort.
- A better solution: Bitonic Sort. 双调排序： O(n) parallel complexity.



Figures from: http://blog.csdn.net/abcjennifer/article/details/43528407

# Master Theorem

```
procedure p( input x of size n ):
  if n < some constant k:
    Solve x directly without recursion
  else:
    Create a subproblems of x, each having size n/b
    Call procedure p recursively on each subproblem
    Combine the results from the subproblems
```

- $T(n) = \underbrace{\boxed{a\,T\left(\frac{n}{b}\right)}}_{\text{Divide}} + \underbrace{\boxed{f(n)}}_{\text{Reduce}}$

Figures from: https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)

# Master Theorem

```
procedure p( input x of size n ):
  if n < some constant k:
    Solve x directly without recursion
  else:
    Create a subproblems of x, each having size n/b
    Call procedure p recursively on each subproblem
    Combine the results from the problems
```

- $T(n) = \underbrace{\boxed{a\,T\left(\frac{n}{b}\right)}}_{\text{Divide}} + \underbrace{\boxed{f(n)}}_{\text{Reduce}}$

**Case 1**

If $f(n) = \Theta\left(n^c\right)$ where $c < \log_b a$ (using Big O notation)
then:

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

**Case 2**

If it is true, for some constant $k \geq 0$, that:

$$f(n) = \Theta\left(n^c \log^k n\right) \text{ where } c = \log_b a$$

then:

$$T(n) = \Theta\left(n^c \log^{k+1} n\right)$$

**Case 3**

If it is true that:

$$f(n) = \Theta\left(n^c\right) \text{ where } c > \log_b a$$

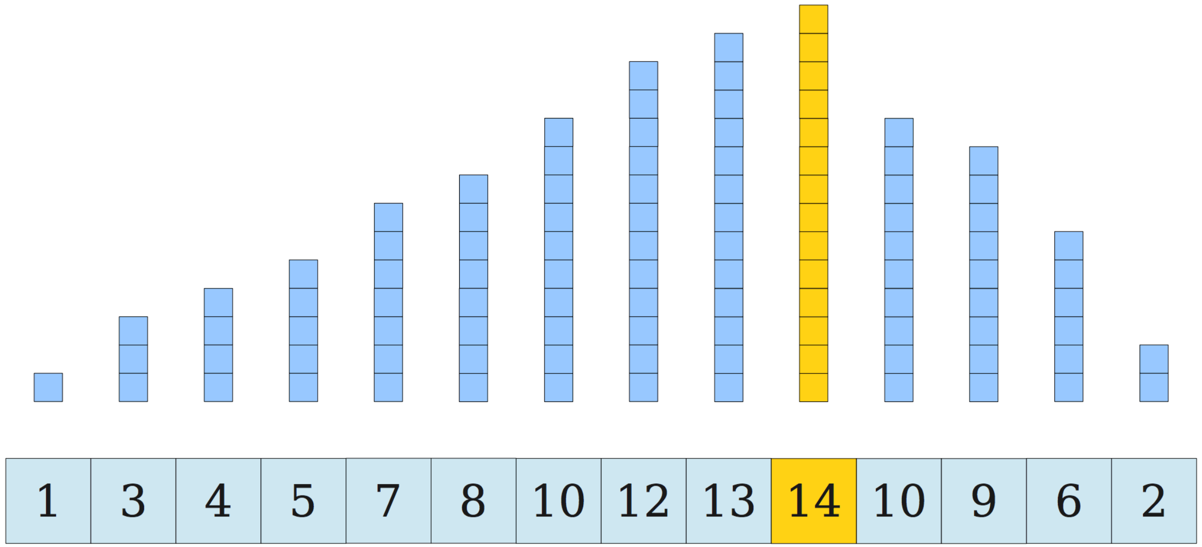
then:

$$T(n) = \Theta\left(f(n)\right)$$

Figures from: https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)

# Warmup: Maximizing Unimodal Arrays

- Unimodal Arrays: = 双调序列。
- 如何找到最大值?




Figures from: Stanford CS161

# Warmup: Maximizing Unimodal Arrays

- Unimodal Arrays: = 双调序列。
- 如何找到最大值?

- 最基本的二分算法开始:
- 问题?

```
procedure unimodalMax(list A, int low, int high):
  if low = high - 1:
     return A[low]

  let mid = ⌊(high + low) / 2⌋
  if A[mid] < A[mid + 1]
     return unimodalMax(A, mid + 1, high)
  else:
     return unimodalMax(A, low, mid + 1)
```

# Warmup: Maximizing Unimodal Arrays

- Unimodal Arrays: = 双调序列。
- 如何找到最大值?

- 最基本的二分算法开始:
- 问题：无法处理Weak-Unimodal

```
procedure unimodalMax(list A, int low, int high):
  if low = high - 1:
     return A[low]

  let mid = ⌊(high + low) / 2⌋
  if A[mid] < A[mid + 1]
     return unimodalMax(A, mid + 1, high)
  else:
     return unimodalMax(A, low, mid + 1)
```

# Warmup: Maximizing Unimodal Arrays

- Unimodal Arrays: = 双调序列。
- 如何找到最大值?

- 最基本的二分算法开始:
- 问题：无法处理Weak-Unimodal。
- 新算法的复杂度?

- 三分法!

```
procedure weakUnimodalMax(list A, int low, int high):
  if low = high - 1:
     return A[low]

  let mid = ⌊(high + low) / 2⌋
  if A[mid] < A[mid + 1]
     return weakUnimodalMax(A, mid + 1, high)
  else if A[mid] > A[mid + 1]
     return weakUnimodalMax(A, low, mid + 1)
  else
     return max(weakUnimodalMax(A, low, mid + 1)
                weakUnimodalMax(A, mid + 1, high))
```

# WarmUp: Common Algorithms

- Selection:
  - Select the k-th element from an unsorted array.
  - $O(n)$ implementation.
  - Application: find the median.
- A subroutine: partition:
  - Partition an array into two parts, s.t. $\forall x \in L, y \in R: x < y$.
  - Select the pivot.

# WarmUp: Pivot Picking (median of medians algorithm)

- 找中位数算法FindMedian：
  - 把数组分成N/5份。找到每一份的中位数。
  - 递归调用中位数算法FindMedian，找到“中位数的中位数”
  - 用该中位数作为主元。

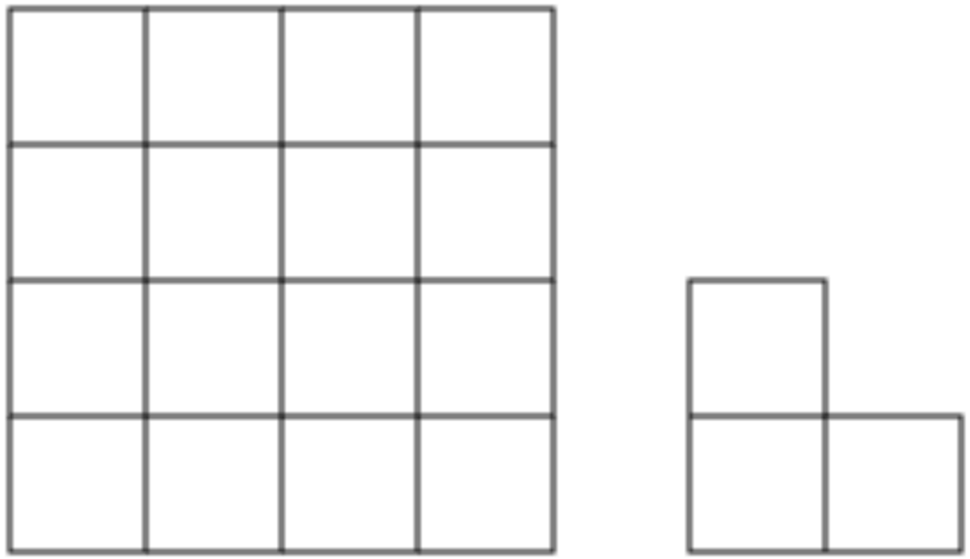
  - 中位数大于1/2的中位数，因此大于~3/10的总数。
  - $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n).$

# WarmUp: Pivot Picking (median of medians algorithm)

- Too complicated???
- Algorithm by:
  - Manuel Blum (Turing Award Winner)
  - Robert Floyd (Turing Award Winner)
  - Vaughan Pratt (Stanford Professor Emeritus)
  - Ron Rivest (Turing Award Winner)
  - Robert Tarjan (Turing Award Winner)
- QuickSelect: Use random selection.
  - Intuition: probability of triggerring worst-case: $\frac{2^{n-1}}{n!}$.

# Tiling with Triominoes

- 给定一个N*N (N = 2^k) 的矩阵，使用3格的骨牌进行覆盖。
- 问最优方案里，有多少的位置不能被覆盖到?
- 覆盖方案是什么?



# Integer Multiplication

- 给定两个长度为N的整数。求他们的乘积。
- A naïve implementation: $O(n^2)$.

# Integer Multiplication

- 给定两个长度为N的整数。求他们的乘积。

- $X = a \cdot 10^{\left[\frac{n}{2}\right]} + b;\ \ Y = c \cdot 10^{\left[\frac{n}{2}\right]} + d\ .$
- $X \cdot Y = (ac) \cdot 10^{2\left[\frac{n}{2}\right]} + (ad + bc) \cdot 10^{\left[\frac{n}{2}\right]} + bd\ .$

- How to compute $ac, (ad + bc), bd$?

# Integer Multiplication

- 给定两个长度为N的整数。求他们的乘积。

- $X = a \cdot 10^{\left\lceil \frac{n}{2} \right\rceil} + b;\ \ Y = c \cdot 10^{\left\lceil \frac{n}{2} \right\rceil} + d$ .
- $X \cdot Y = (ac) \cdot 10^{2\left\lceil \frac{n}{2} \right\rceil} + (ad + bc) \cdot 10^{\left\lceil \frac{n}{2} \right\rceil} + bd$ .

- How to compute $ac, (ad + bc), bd$?
- $E = ac; F = bd; G = (a + b)(c + d); \Longrightarrow (ad + bc) = G - E - F$.
- Complexity: $O\left(n^{\log_2 3}\right) \approx O(n^{1.585})$. FFT: $O(n \log^3 n)$. Fürer‘s algorithm (2008): $O\left(n \log n\, 2^{\log^* n}\right)$.

# Similar Trick: Strassen Algorithm

- Matrix multiplication.
- An naïve implementation: $O(n^3)$.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

Figures from: https://en.wikipedia.org/wiki/Strassen_algorithm

# Similar Trick: Strassen Algorithm

- Matrix multiplication.
- An naïve implementation: $O(n^3)$.
- Strassen Algorithm: $O\left(n^{\log_2 7}\right) \approx O(n^{2.807})$.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{1,2} &= \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\
\mathbf{C}_{2,1} &= \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{2,2} &= \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}
\end{aligned}$$

$$\begin{aligned}
\mathbf{M}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{M}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{M}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{M}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{M}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{M}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{M}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{1,2} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{2,1} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{2,2} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}$$

# Warmup: CF 888E

- 给定$n$个正整数$a_i$和一个模数$m$。找到一个子集$B$，最小化：
- $(\sum_{b \in B} b) \bmod m$.
- $n \le 35. a_i, m \le 10^9$.

# Warmup: CF 868F

- 给一个长度为$n$的序列$a_i$，现在要求分成$k$块，每一块的代价是“值相同但是位置不同（无序）数对个数”。求最小总代价。
- $n \leq 10^5, 2 \leq k \leq \min(n, 20), a_i \leq n.$

# Warmup: CF 868F

- 给一个长度为$n$的序列$a_i$，现在要求分成$k$块，每一块的代价是“值相同但是位置不同（无序）数对个数”。求最小总代价。
- $n \le 10^5, 2 \le k \le \min(n, 20), a_i \le n.$
- $f_{i,k} = \min\{f_{j,k-1} + cost(j+1, i)\}$

# BZOJ 3263: 三元逆序对

- 给定$n$个三元组 $(A_i, B_i, C_i)$ .
- 找出所有的 $(i, j)$, s.t. $A_i < A_j, \mathrm{B_i} < \mathrm{B_j}, \mathrm{C_i} < \mathrm{C_j}$ .
- $n \le 10^5, A_i, B_i, C_i \le 2 * 10^5$.

# BZOJ 3263: 三元逆序对

- 给定$n$个三元组 $(A_i, B_i, C_i)$ .
- 找出所有的 $(i, j)$, s.t. $A_i < A_j, \mathrm{B_i} < \mathrm{B_j}, \mathrm{C_i} < \mathrm{C_j}$ .
- $n \le 10^5, A_i, B_i, C_i \le 2 * 10^5$.

- 按照$A_i$ 排序。考虑对长度分治。每次考虑$[L, R)$区间内的答案。
- 递归处理$[L, M), [M, R)$.
- 考虑所有跨中间点的数对，满足$i$在$[L, M)$区间，$j$在$[M, R)$区间。
- 我们枚举从小到大$B$，用一个线段树维护$C$。

# BZOJ 3456: 贸易线路

- 求点数为$n$的无向简单图个数。
- 简单图： 无重边，无自环。
- $n \le 10^5$.

# BZOJ 3456: 贸易线路

- 求点数为$n$的无向简单图个数。

- 设$f(n)$表示有$n$个点的有标号简单连通无向图的个数；
- 设g$(n)$表示有$n$个点的有标号简单　　　无向图的个数。
- 我们枚举点I所在的连通块大小，得到：
- $2^{\binom{n}{2}} = g(n) = \sum_{i=1}^{n}\binom{n-1}{i-1} * f(i) * g(n-i)$.

# BZOJ 3456: 贸易线路

- 求点数为$n$的无向简单图个数。

- $2^{\binom{n}{2}} = g(n) = \sum_{i=1}^{n} \binom{n-1}{i-1} * f(i) * g(n-i)$.
- $\frac{2^{\binom{n}{2}}}{(n-1)!} = \sum_{i=1}^{n} \frac{f(i)}{(i-1)!} * \frac{g(n-i)}{(n-i)!}$.
- $f(n) = 2^{\binom{n}{2}} - \sum_{i=1}^{n-1} \binom{n-1}{i-1} 2^{\binom{n}{k}} f(i)$.

- 练习题：ZOJ 3874 Permutation Graph

# Balkan OI 2007: Mokia

- 有一个W *W的棋盘，每个格子内有一个数，初始的时候全部为0。
- 现在要求维护两种操作：
  - Add：将格子(x, y)内的数加上A。
  - Query：询问矩阵(x0，y0，x1，y1)内所有格子的数的和。
- $W \leq 2 * 10^5, Q \leq 10^5$.

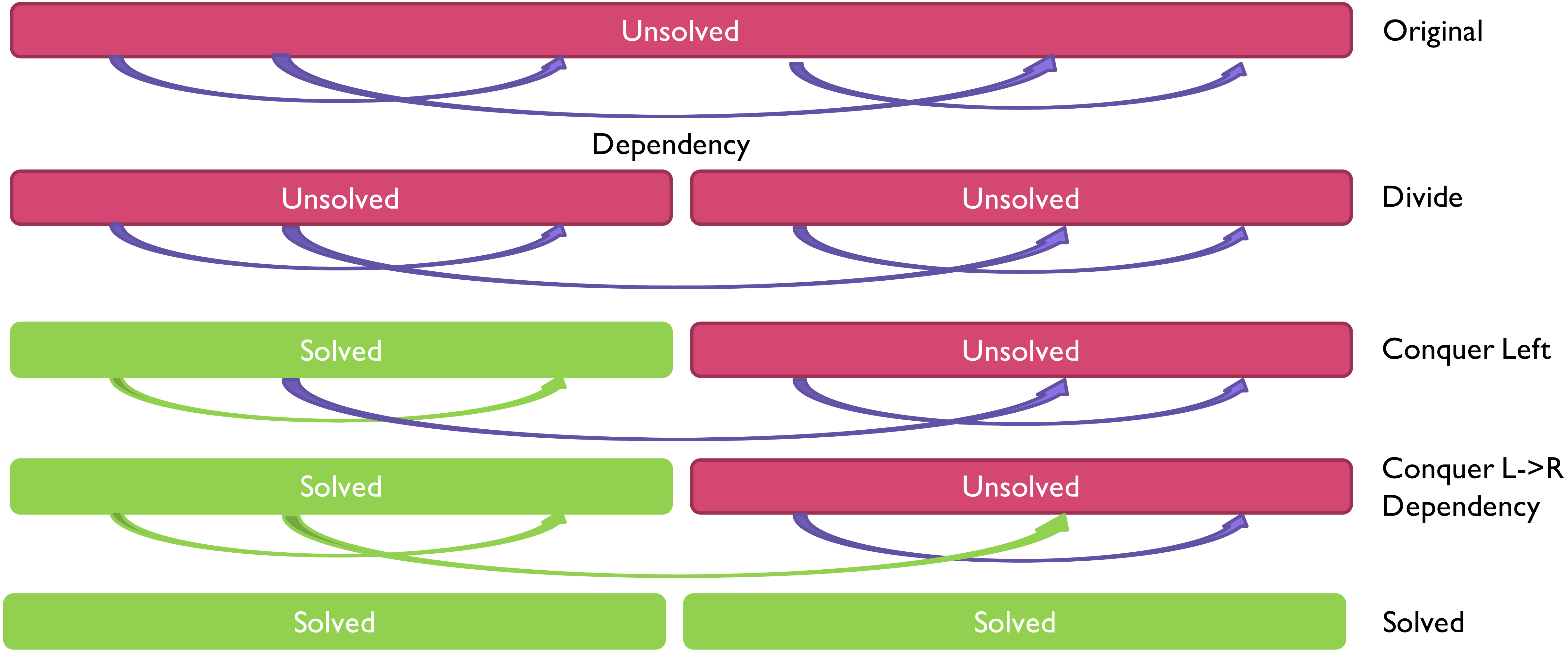
- 二维线段树?

# Balkan OI 2007: Mokia

- 只需要考虑：前一半操作对后一半查询产生的影响。

- 问题转变为：给定点集$(x, y, A)$，查询若干个矩形内的点。
- 扫描线做法，按照$x$坐标排序。对$y$坐标维护线段树。

# CDQ 分治


Unsolved
Original
Dependency
Unsolved
Unsolved
Divide
Solved
Unsolved
Conquer Left
Solved
Unsolved
Conquer L->R Dependency
Solved
Solved
Solved

# BZOJ 2716: 天使玩偶

- 维护一个二维平面，支持如下两种操作：
  - 插入一个点$(x,y)$;
  - 询问距离一个点$(x_q,y_q)$最近的点，曼哈顿距离。
- $|x,y| \leq 10^5, q \leq 10^5$.

- 动态K-d树?

# BZOJ 2716: 天使玩偶

- $dis(i,q) = |x_i - \mathrm{x}_q| + |y_i - y_q|$.
- 分4种情况讨论，先解决$dis_1(i,q) = (x_i - \mathrm{x}_q) + (y_i - y_q)$。
- 求满足：$\mathrm{x_i} > x_q; y_i > y_q$; 且$x_i + \mathrm{y}_i$最小的点。

- 带权为$x_i + \mathrm{y}_i$的点集，二维区间最值。

# NOI 2007: 货币兑换

- 有两种股票。你有$n$天时间进行货币买卖。第i天你可以进行两种操作:
  - 给定一个数$s$，表示花费的现金。按照市值$a_i, b_i$收获两种股票，两种股票的比例是固定的$r_i$（即你不可以选择只买A不买B）。
  - 给定一个比例$k$，按照市值$a_i, b_i$把手中$k\%$的持有的A和B卖掉。（即你不可以选择只卖A不卖B）。
- 给定所有参数，问n天后你的最大持有现金数（最后一天必须全部卖掉）。

# NOI 2007: 货币兑换

- 有两种股票。你有$n$天时间进行货币买卖。第i天你可以进行两种操作：
  - 给定一个数$s$，表示花费的现金。按照市值$a_i, b_i$收获两种股票，两种股票的比例是固定的$r_i$（即你不可以选择只买A不买B）。
  - 给定一个比例$k$，按照市值$a_i, b_i$把手中$k$%的持有的A和B卖掉。（即你不可以选择只卖A不卖B）。

- 显然为了最大化利益，每次都会操作所有的现金/股票。
- 令$F[i]$表示第$i$天，最多能持有F[$i$]的现金。
- 枚举上一个买入交易日$j$，我们在第$j$天花所有钱买入股票，在第$i$天全部卖掉。
- $F[i] = \max\{F[i-1], fa(j) * a[i] + fb(j) * b[j]\}$
- $fb[j] = \frac{F[j]}{(a[j]*r[j]+b[j])}; fa[j] = \frac{a[j]*F[j]}{(a[j]*r[j]+b[j])}.$

# NOI 2007: 货币兑换

- $F[i] = \max\{F[i-1], fa(j) * a[i] + fb(j) * b[j]\}$
- $fb[j] = \frac{F[j]}{(a[j]*r[j]+b[j])}; fa[j] = \frac{a[j]*F[j]}{(a[j]*r[j]+b[j])}.$
- 对于$i$，决策$j$优于$k$的条件是什么？
- $\frac{r[j]*F[j]-r[k]*F[k]}{F[j]-F[k]} > -\frac{b[i]}{a[i]}.$
- 需要的操作：
  - 加点，维护上凸壳
  - 用一条固定斜率的线从上到下扫描，回答碰到的第一个点。=>斜率的二分。

# NOI 2007: 货币兑换

- 需要的操作：
  - 加点，维护上凸壳
  - 用一条固定斜率的线从上到下扫描，回答碰到的第一个点。=>斜率的二分。

- 使用CDQ分治，操作变为：
  - 给定点集，构造凸包。
  - 给定凸包，二分斜率。
- 进一步优化?

- 练习题WF2011, Machine Works

# POI2011: METEOR

- 有$m$个天文台排成一个环形，每个天文台属于$n$个国家中的一个。
- 有$k$场流星雨，第$i$场波及范围为$[l[i], r[i]]$，为其中每一个天文台提供$a_i$的陨石。
- 每个国家有陨石需求量$b_j$。
- 输出每一个国家在第几场流星雨后，能达到陨石需求。

- 线段树可以维护流星雨的信息。
- 怎么统计呢？ O(国家拥有的天文台)?

# POI2011: METEOR

- 有$k$场流星雨，第$i$场波及范围为$\left[l[i],r[i]\right]$，为其中每一个天文台提供$\mathrm{a}_i$的陨石。
- 每个国家有陨石需求量$b_j$。

- 考虑对答案“分治”。
- 每次询问加上$[L,M)$这段区间的流星，有多少国家满足需求了?
- 满足的，二分左侧区间。不满足的，二分右侧区间。
- 右侧区间信息怎么维护？主席树/撤销操作/简单统计。

# BZOJ 3110: K大数查询

- 有$n$个vector：$v_i$排成一排，维护如下两种操作：
  - 1 a b c，对$i \in [a,b], v_i.pushback(c)$.
  - 2 a b c，查询$\bigcup_{i\in[a,b]} v_i$集合的第$c$大元素。

# BZOJ 3110: K大数查询

- 有$n$个vector：$v_i$排成一排，维护如下两种操作：
  - 1 a b c，对$i \in [a, b], v_i.pushback(c)$.
  - 2 a b c，查询$\cup_{i\in[a,b]} v_i$集合的第$c$大元素。

- 传统做法：树套树。
- 我们对答案“分治”。那么每次要求的就是对于每一个询问$[a, b] \rightarrow ans$。
- 扫描所有操作，每次询问是否存在至少$c$个数，他们$< M$（二分的$M$），且在$[a, b]$区间范围内。

# Tsinsen A1333: 矩阵乘法

- 给定$n * n$的矩阵，每次询问一个二维区间内的第$k$小值。
- $n \leq 500; q \leq 60000.$

# Tsinsen A1333: 矩阵乘法

- 给定$n * n$的矩阵，每次询问一个二维区间内的第$k$小值。

- 对答案“分治”。每次处理的是值范围在[L, M]区间的数，有多少个在每个Query的矩形里。
- 二维树状数组。
- 优化：扫描线，用一维树状数组维护。

# K近邻相交查询

- 给定N个排序的实数，有若干组询问：
  - (x, y, k)，询问距离x最近（差的绝对值最小）的k个点，和距离y最近的k个点，是否有交集。
- N, K <= 10^5, Ai <= 10^9.

# K近邻相交查询

- 二维情况。
- 给定N个平面上的点(xi, yi)，每次询问：
  - (x, y, k)，询问距离x最近（曼哈顿距离，即坐标差的绝对值之和）的k个点，和距离y最近的k个点，是否有交集。
- N, K <= 10^5, xi, yi <= 10^9.

# NOI 2012 骑行川藏

# NOI 2012 骑行川藏

求一组实数$x_i$使得

$$\sum \mathrm{k_i} * s_i * (v_i - x_i)^2 = E$$

且$\sum \mathrm{s_i}/x_i$最小

$k_{\mathrm{i}}, v_i, s_i, E$都是给定的常数

# NOI 2012 骑行川藏

定义$f'(x_i)$表示多元函数 $f(x)$关于变量$x_i$的偏导

在此问题中，$f(x)=\sum\frac{s_i}{x_i}$

约束条件$g(x) = \sum k_i * s_i * (v_i - x_i)^2$

$$f'(x_i) = \frac{\left(f(x+\Delta x) - f(x)\right)}{\Delta x} = \frac{\frac{s_i}{x+\Delta x} - \frac{s_i}{x}}{\Delta x} = -\frac{s_i}{x_i^2}$$

$$g'(x_i) = \frac{\left(g(x+\Delta x) - g(x)\right)}{\Delta x} = 2 * s_i * k_i * (x_i - v_i)$$

于是我们得到方程组 $-\frac{s_i}{x_i^2} = \lambda * 2 * s_i * k_i * (x_i - v_i)$

以及$\sum k_i * s_i * (v_i - x_i)^2 = E$

观察发现，$\lambda(<0)$的绝对值越小，$x_i$就越大，$g(x)$就越大

就是说，$g(x)$是关于$\lambda$单调的，也就是说我们可以二分$\lambda$的值

来解这个方程组