# Chapter 13

# Introduction to Randomized Algorithms: Quick Sort and Quick Selection

**CS 473: Fundamental Algorithms, Spring 2011**
March 8, 2011

## 13.1 Introduction to Randomized Algorithms

## 13.2 Introduction

#### 13.2.0.1 Randomized Algorithms

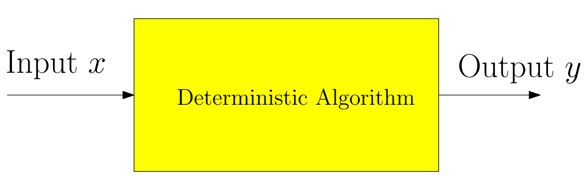
#### 13.2.0.2 Example: Randomized QuickSort

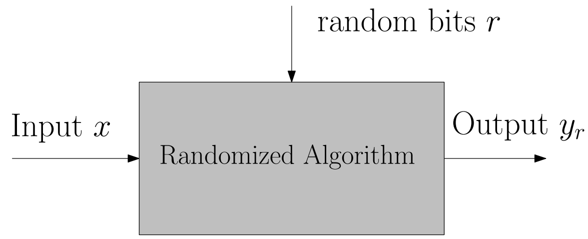
**QuickSort** [Hoare, 1962]

(A) Pick a pivot element from array
(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Recursively sort the subarrays, and concatenate them.

**Randomized QuickSort**

(A) Pick a pivot element *uniformly at random* from the array






(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Recursively sort the subarrays, and concatenate them.

#### 13.2.0.3 Example: Randomized Quicksort

Recall: **QuickSort** can take $\Omega(n^2)$ time to sort array of size $n$.

**Theorem 13.2.1** *Randomized* **QuickSort** *sorts a given array of length* $n$ *in* $O(n \log n)$ expected *time.*

**Note:** On *every* input randomized **QuickSort** takes $O(n \log n)$ time in expectation. On *every* input it may take $\Omega(n^2)$ time with some small probability.

#### 13.2.0.4 Example: Verifying Matrix Multiplication

**Problem**
Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Deterministic algorithm:
(A) Multiply $A$ and $B$ and check if equal to $C$.
(B) Running time? $O(n^3)$ by straight forward approach. $O(n^{2.37})$ with fast matrix multiplication (complicated and impractical).

#### 13.2.0.5 Example: Verifying Matrix Multiplication

**Problem**
Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Randomized algorithm:
(A) Pick a random $n \times 1$ vector $r$.
(B) Return the answer of the equality $ABr = Cr$.
(C) Running time? $O(n^2)$!

**Theorem 13.2.2** *If* $AB = C$ *then the algorithm will always say YES. If* $AB \neq C$ *then the algorithm will say YES with probability at most* $1/2$. *Can repeat the algorithm* 100 *times independently to reduce the probability of a false positive to* $1/2^{100}$.

#### 13.2.0.6 Why randomized algorithms?

(A) Many many applications in algorithms, data structures and computer science!
(B) In some cases only known algorithms are randomized or randomness is provably necessary.
(C) Often randomized algorithms are (much) simpler and/or more efficient.
(D) Several deep connections to mathematics, physics etc.
(E) ...
(F) Lots of fun!

#### 13.2.0.7 Where do I get random bits?

**Question:** Are true random bits available in practice?
(A) Buy them!
(B) CPUs use physical phenomena to generate random bits.
(C) Can use pseudo-random bits or semi-random bits from nature. Several fundamental unresolved questions in complexity theory on this topic. Beyond the scope of this course.
(D) In practice pseudo-random generators work quite well in many applications.
(E) The model is interesting to think in the abstract and is very useful even as a theoretical construct. One can *derandomize* randomized algorithms to obtain deterministic algorithms.

#### 13.2.0.8 Average case analysis vs Randomized algorithms

**Average case analysis:**
(A) Fix a deterministic algorithm.
(B) Assume inputs comes from a probability distribution.
(C) Analyze the algorithm's *average* performance over the distribution over inputs.
**Randomized algorithms:**
(A) Algorithm uses random bits in addition to input.
(B) Analyze algorithms *average* performance over the given input where the average is over the random bits that the algorithm uses.
(C) On each input behaviour of algorithm is random. Analyze worst-case over all inputs of the (average) performance.

## 13.3 Basics of Discrete Probability

#### 13.3.0.9 Discrete Probability

We restrict attention to finite probability spaces.

**Definition 13.3.1** *A discrete probability space is a pair* $(\Omega, \mathbf{Pr})$ *consists of finite set* $\Omega$ *of* elementary *events and function* $p : \Omega \to [0, 1]$ *which assigns a probability* $\mathbf{Pr}[\omega]$ *for each*

$\omega \in \Omega$ *such that* $\sum_{\omega\in\Omega} \mathbf{Pr}[\omega] = 1$.

**Example 13.3.2** *An unbiased coin.* $\Omega = \{H, T\}$ *and* $\mathbf{Pr}[H] = \mathbf{Pr}[T] = 1/2$.

**Example 13.3.3** *A* 6*-sided unbiased die.* $\Omega = \{1, 2, 3, 4, 5, 6\}$ *and* $\mathbf{Pr}[i] = 1/6$ *for* $1 \le i \le 6$.

## 13.3.1 Discrete Probability

### 13.3.1.1 And more examples

**Example 13.3.4** *A biased coin.* $\Omega = \{H, T\}$ *and* $\mathbf{Pr}[H] = 2/3, \mathbf{Pr}[T] = 1/3$.

**Example 13.3.5** *Two independent unbiased coins.* $\Omega = \{HH, TT, HT, TH\}$ *and* $\mathbf{Pr}[HH] = \mathbf{Pr}[TT] = \mathbf{Pr}[HT] = \mathbf{Pr}[TH] = 1/4$.

**Example 13.3.6** *A pair of (highly) correlated dice.*
$\Omega = \{(i, j) \mid 1 \le i \le 6, 1 \le j \le 6\}$.
$\mathbf{Pr}[i, i] = 1/6$ *for* $1 \le i \le 6$ *and* $\mathbf{Pr}[i, j] = 0$ *if* $i \ne j$.

### 13.3.1.2 Events

**Definition 13.3.7** *Given a probability space* $(\Omega, \mathbf{Pr})$ *an* **event** *is a subset of* $\Omega$. *In other words an event is a collection of elementary events. The probability of an event* $A$*, denoted by* $\mathbf{Pr}[A]$*, is* $\sum_{\omega\in A} \mathbf{Pr}[\omega]$. *The complement of an event* $A \subseteq \Omega$ *is the event* $\Omega \setminus A$ *frequently denoted by* $\bar{A}$.

## 13.3.2 Events

### 13.3.2.1 Examples

**Example 13.3.8** *A pair of independent dice.* $\Omega = \{(i, j) \mid 1 \le i \le 6, 1 \le j \le 6\}$.

*(A)* *Let* $A$ *be the event that the sum of the two numbers on the dice is even. Then* $A = \{(i, j) \in \Omega \mid (i + j)$ *is even*$\}$. $\mathbf{Pr}[A] = |A|/36 = 1/2$.

*(B)* *Let* $B$ *be the event that the first die has* 1*. Then* $B = \Big\{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6)\Big\}$. $\mathbf{Pr}[B] = 6/36 = 1/6$.

### 13.3.2.2 Independent Events

**Definition 13.3.9** *Given a probability space* $(\Omega, \mathbf{Pr})$ *and two events* $A, B$ *are* **independent** *if and only if* $\mathbf{Pr}[A \cap B] = \mathbf{Pr}[A]\,\mathbf{Pr}[B]$. *Otherwise they are* dependent*. In other words* $A, B$ *independent implies one does not affect the other.*

**Example 13.3.10** *Two coins.* $\Omega = \{HH, TT, HT, TH\}$ *and* $\mathbf{Pr}[HH] = \mathbf{Pr}[TT] = \mathbf{Pr}[HT] = \mathbf{Pr}[TH] = 1/4$.

*(A) A is the event that the first coin is heads and B is the event that second coin is tails. $A, B$ are independent.*
*(B) A is the event that the two coins are different. B is the event that the second coin is heads. $A, B$ independent.*

### 13.3.3 Independent Events

#### 13.3.3.1 Examples

**Example 13.3.11** *A is the event that both are not tails and B is event that second coin is heads. $A, B$ are dependent.*

#### 13.3.3.2 Random Variables

**Definition 13.3.12** *Given a probability space $(\Omega, \mathbf{Pr})$ a (real-valued) random variable $X$ over $\Omega$ is a function that maps each elementary event to a real number. In other words $X : \Omega \to \mathbb{R}$.*

**Example 13.3.13** *A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathbf{Pr}[i] = 1/6$ for $1 \leq i \leq 6$.*
*(A) $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$.*
*(B) $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$.*

**Definition 13.3.14** *A* **binary random variable** *is one that takes on values in $\{0, 1\}$.*

#### 13.3.3.3 Indicator Random Variables

Special type of random variables that are quite useful.

**Definition 13.3.15** *Given a probability space $(\Omega, \mathbf{Pr})$ and an event $A \subseteq \Omega$ the indicator random variable $X_A$ is a binary random variable where $X_A(\omega) = 1$ if $\omega \in A$ and $X_A(\omega) = 0$ if $\omega \notin A$.*

**Example 13.3.16** *A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathbf{Pr}[i] = 1/6$ for $1 \leq i \leq 6$. Let A be the even that i is divisible by 3. Then $X_A(i) = 1$ if $i = 3, 6$ and 0 otherwise.*

#### 13.3.3.4 Expectation

**Definition 13.3.17** *For a random variable $X$ over a probability space $(\Omega, \mathbf{Pr})$ the* **expectation** *of $X$ is defined as $\sum_{\omega \in \Omega} \mathbf{Pr}[\omega] X(\omega)$. In other words, the expectation is the average value of $X$ according to the probabilities given by $\mathbf{Pr}[\cdot]$.*

**Example 13.3.18** *A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathbf{Pr}[i] = 1/6$ for $1 \leq i \leq 6$.*
*(A) $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$. Then $\mathbf{E}[X] = 1/2$.*
*(B) $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$. Then $\mathbf{E}[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = 91/6$.*

#### 13.3.3.5 Expectation

**Proposition 13.3.19** *For an indicator variable $X_A$, $\mathbf{E}[X_A] = \mathbf{Pr}[A]$.*

*Proof*:

$$
\begin{aligned}
\mathbf{E}[X_A] &= \sum_{y \in \Omega} X_A(y)\, \mathbf{Pr}[y] \\
&= \sum_{y \in A} 1 \cdot \mathbf{Pr}[y] + \sum_{y \in \Omega \setminus A} 0 \cdot \mathbf{Pr}[y] \\
&= \sum_{y \in A} \mathbf{Pr}[y] \\
&= \mathbf{Pr}[A]\,.
\end{aligned}
$$

■

#### 13.3.3.6 Linearity of Expectation

**Lemma 13.3.20** *Let $X, Y$ be two random variables over a probability space $(\Omega, \mathbf{Pr})$. Then $\mathbf{E}[X + Y] = \mathbf{E}[X] + \mathbf{E}[Y]$.*

*Proof*:

$$
\begin{aligned}
\mathbf{E}[X + Y] &= \sum_{\omega \in \Omega} \mathbf{Pr}[\omega]\,(X(\omega) + Y(\omega)) \\
&= \sum_{\omega \in \Omega} \mathbf{Pr}[\omega]\, X(\omega) + \sum_{\omega \in \Omega} \mathbf{Pr}[\omega]\, Y(\omega) = \mathbf{E}[X] + \mathbf{E}[Y]\,.
\end{aligned}
$$

■

**Corollary 13.3.21** $\mathbf{E}[a_1 X_1 + a_2 X_2 + \ldots + a_n X_n] = \sum_{i=1}^{n} a_i\, \mathbf{E}[X_i]$.

## 13.4 Analyzing Randomized Algorithms

#### 13.4.0.7 Types of Randomized Algorithms

Typically one encounters the following types:

(A) ***Las Vegas randomized algorithms:*** for a given input $x$ output of algorithm is always correct but the running time is a random variable. In this case we are interested in analyzing the *expected* running time.

(B) ***Monte Carlo randomized algorithms:*** for a given input $x$ the running time is deterministic but the output is random; correct with some probability. In this case we are interested in analyzing the *probability* of the correct output (and also the running time).

(C) Algorithms whose running time and output may both be random.

#### 13.4.0.8 Analyzing Las Vegas Algorithms

*Deterministic* algorithm $Q$ for a problem $\Pi$:
(A) Let $Q(x)$ be the time for $Q$ to run on input $x$ of length $|x|$.
(B) Worst-case analysis: run time on worst input for a given size $n$.

$$T_{wc}(n) = \max_{x:|x|=n} Q(x).$$

*Randomized* algorithm $R$ for a problem $\Pi$:
(A) Let $R(x)$ be the time for $Q$ to run on input $x$ of length $|x|$.
(B) $R(x)$ is a random variable: depends on random bits used by $R$.
(C) $\mathbf{E}[R(x)]$ is the expected running time for $R$ on $x$
(D) Worst-case analysis: expected time on worst input of size $n$

$$T_{rand-wc}(n) = \max_{x:|x|=n} \mathbf{E}[Q(x)].$$

#### 13.4.0.9 Analyzing Monte Carlo Algorithms

*Randomized* algorithm $M$ for a problem $\Pi$:
(A) Let $M(x)$ be the time for $M$ to run on input $x$ of length $|x|$. For Monte Carlo, assumption is that run time is deterministic.
(B) Let $\mathbf{Pr}[x]$ be the probability that $M$ is correct on $x$.
(C) $\mathbf{Pr}[x]$ is a random variable: depends on random bits used by $M$.
(D) Worst-case analysis: success probability on worst input

$$P_{rand-wc}(n) = \min_{x:|x|=n} \mathbf{Pr}[x].$$

## 13.5 Randomized Quick Sort and Selection

## 13.6 Randomized Quick Sort

#### 13.6.0.10 Randomized QuickSort

**Randomized QuickSort**
(A) Pick a pivot element *uniformly at random* from the array
(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Recursively sort the subarrays, and concatenate them.

#### 13.6.0.11 Example

(A) array: 16, 12, 14, 20, 5, 3, 18, 19, 1

### 13.6.0.12 Analysis via Recurrence

(A) Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized **QuickSort** on $A$.
(B) Note that $Q(A)$ is a random variable
(C) Let $A^i_{\text{left}}$ and $A^i_{\text{right}}$ be the left and right arrays obtained if:

pivot is of rank $i$ in $A$.

$$Q(A) = n + \sum_{i=1}^{n} \mathbf{Pr}[\text{pivot has rank } i] \left( Q(A^i_{\text{left}}) + Q(A^i_{\text{right}}) \right)$$

Since each element of $A$ has probability exactly of $1/n$ of being chosen:

$$Q(A) = n + \sum_{i=1}^{n} \frac{1}{n} \left( Q(A^i_{\text{left}}) + Q(A^i_{\text{right}}) \right)$$

### 13.6.0.13 Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} \mathbf{E}[Q(A)]$ be the worst-case expected running time of randomized **QuickSort** on arrays of size $n$.

We have, for any $A$:

$$Q(A) = n + \sum_{i=1}^{n} \mathbf{Pr}[\text{pivot has rank } i] \left( Q(A^i_{\text{left}}) + Q(A^i_{\text{right}}) \right)$$

Therefore, by linearity of expectation:

$$\mathbf{E}\Big[Q(A)\Big] = n + \sum_{i=1}^{n} \mathbf{Pr}[\text{pivot of rank } i] \Big( \mathbf{E}\big[Q(A^i_{\text{left}})\big] + \mathbf{E}\big[Q(A^i_{\text{right}})\big] \Big).$$

$$\Rightarrow \quad \mathbf{E}\Big[Q(A)\Big] \leq n + \sum_{i=1}^{n} \frac{1}{n} \left( T(i-1) + T(n-i) \right).$$

### 13.6.0.14 Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} \mathbf{E}[Q(A)]$ be the worst-case expected running time of randomized **QuickSort** on arrays of size $n$.

We derived:

$$\mathbf{E}[Q(A)] \leq n + \sum_{i=1}^{n} \frac{1}{n} \left( T(i-1) + T(n-i) \right).$$

Note that above holds for any $A$ of size $n$. Therefore

$$\max_{A:|A|=n} \mathbf{E}[Q(A)] = T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n} \left( T(i-1) + T(n-i) \right).$$

#### 13.6.0.15 Solving the Recurrence

$$T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n}\left(T(i-1) + T(n-i)\right)$$

with base case $T(1) = 0$.

**Lemma 13.6.1** $T(n) = O(n \log n)$.

*Proof*: (Guess and) Verify by induction. ■

#### 13.6.0.16 A Slick Analysis of QuickSort

Let $Q(A)$ be number of comparisons done on input array $A$:

(A) For $1 \leq i < j < n$ let $R_{ij}$ be the event that rank $i$ element is compared with rank $j$ element.

(B) $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise 0.

$$Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$$

and hence by linearity of expectation,

$$\mathbf{E}\Big[Q(A)\Big] = \sum_{1 \leq i < j \leq n} \mathbf{E}[X_{ij}] = \sum_{1 \leq i < j \leq n} \mathbf{Pr}[R_{ij}].$$

#### 13.6.0.17 A Slick Analysis of QuickSort

**Question:** What is $\mathbf{Pr}[R_{ij}]$?

**Lemma 13.6.2** $\mathbf{Pr}[R_{ij}] = \frac{2}{(j-i+1)}$.

*Proof*: Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be elements of $A$ in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

**Observation:** If pivot is chosen outside $S$ then all of $S$ either in left array or right array.

**Observation:** $a_i$ and $a_j$ separated when a pivot is chosen from $S$ for the first time. Once separated no comparison.

**Observation:** $a_i$ is compared with $a_j$ if and only if either $a_i$ or $a_j$ is chosen as a pivot from $S$ at separation... ■

### 13.6.1 A Slick Analysis of QuickSort

#### 13.6.1.1 Continued...

**Lemma 13.6.3** $\mathbf{Pr}[R_{ij}] = \frac{2}{(j-i+1)}$.

*Proof*: Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be sort of $A$. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

**Observation:** $a_i$ is compared with $a_j$ if and only if either $a_i$ or $a_j$ is chosen as a pivot from $S$ at separation.

**Observation:** Given that pivot is chosen from $S$ the probability that it is $a_i$ or $a_j$ is exactly $2/|S| = 2/(j-i+1)$ since the pivot is chosen uniformly at random from the array. ∎

### 13.6.2 A Slick Analysis of QuickSort

#### 13.6.2.1 Continued...

$$\mathbf{E}\Big[Q(A)\Big] = \sum_{1 \le i < j \le n} \mathbf{E}[X_{ij}] = \sum_{1 \le i < j \le n} \mathbf{Pr}[R_{ij}] \, .$$

**Lemma 13.6.4** $\mathbf{Pr}[R_{ij}] = \frac{2}{(j-i+1)}$.

$$\begin{aligned}
\mathbf{E}\Big[Q(A)\Big] &= \sum_{1 \le i < j \le n} \mathbf{Pr}[R_{ij}] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = 2\sum_{i=1}^{n-1} \sum_{i<j}^{n} \frac{1}{j-i+1} \\
&= 2\sum_{i=1}^{n-1} (H_{n-i+1} - 1) \le 2 \sum_{1 \le i < n} H_n \\
&\le 2nH_n = O(n \log n)
\end{aligned}$$

## 13.7 Randomized Selection

#### 13.7.0.2 Randomized Quick Selection

**Input** Unsorted array $A$ of $n$ integers

**Goal** Find the $j$th smallest number in $A$ (*rank* $j$ number)

**Randomized Quick Selection**

(A) Pick a pivot element *uniformly at random* from the array
(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Return pivot if rank of pivot is $j$
(D) Otherwise recurse on one of the arrays depending on $j$ and their sizes.

#### 13.7.0.3 Algorithm for Randomized Selection

```
QuickSelect(A, j):
    Pick pivot x uniformly at random fro
    Partition A into A_less, x, and A_greate
    if (|A_less| = j − 1) then
        return x
    if (|A_less|) ≥ j) then
        return QuickSelect(A_less, j)
    else
        return QuickSelect(A_greater, j − |
```

**Assume** for simplicity that $A$ has distinct elements.

#### 13.7.0.4 Analysis via Recurrence

(A) Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized selection on $A$ for selecting rank $j$ element.
(B) Note that $Q(A)$ is a random variable
(C) Let $A^i_{\text{less}}$ and $A^i_{\text{greater}}$ be the left and right arrays obtained if pivot is rank $i$ element of $A$.
(D) Algorithm recurses on $A^i_{\text{less}}$ if $j < i$ and recurses on $A^i_{\text{greater}}$ if $j > i$ and terminates if $j = i$.

$$\begin{aligned} Q(A) &= n + \sum_{i=1}^{j-1} \mathbf{Pr}[\text{pivot has rank } i]\, Q(A^i_{\text{greater}}) \\ &\quad + \sum_{i=j+1}^{n} \mathbf{Pr}[\text{pivot has rank } i]\, Q(A^i_{\text{less}}) \end{aligned}$$

#### 13.7.0.5 Analyzing the Recurrence

As in **QuickSort** we obtain the following recurrence where $T(n)$ is the worst-case expected time.

$$T(n) \le n + \frac{1}{n}\left(\sum_{i=1}^{j-1} T(n-i) + \sum_{i=j}^{n} T(i-1)\right).$$

**Theorem 13.7.1** $T(n) = O(n)$.

*Proof*: (Guess and) Verify by induction (see next slide). ∎

#### 13.7.0.6 Analyzing the recurrence

**Theorem 13.7.2** $T(n) = O(n)$.

Prove by induction that $T(n) \le \alpha n$ for some constant $\alpha \ge 1$ to be fixed later.
**Base case:** $n = 1$, we have $T(1) = 0$ since no comparisons needed and hence $T(1) \le \alpha$.

**Induction step:** Assume $T(k) \leq \alpha k$ for $1 \leq k < n$ and prove it for $T(n)$. We have by the recurrence:

$$\begin{aligned} T(n) &\leq n + \frac{1}{n}(\sum_{i=1}^{j-1} T(n-i) + \sum_{i=j^n} T(i-1)) \\ &\leq n + \frac{\alpha}{n}(\sum_{i=1}^{j-1}(n-i) + \sum_{i=j}^{n}(i-1)) \quad \text{by applying induction} \end{aligned}$$

#### 13.7.0.7 Analyzing the recurrence

$$\begin{aligned} T(n) &\leq n + \frac{\alpha}{n}(\sum_{i=1}^{j-1}(n-i) + \sum_{i=j}^{n}(i-1)) \\ &\leq n + \frac{\alpha}{n}((j-1)(2n-j)/2 + (n-j+1)(n+j-2)/2) \\ &\leq n + \frac{\alpha}{2n}(n^2 + 2nj - 2j^2 - 3n + 4j - 2) \\ &\quad \text{above expression maximized when } j = (n+1)/2\text{: calculus} \\ &\leq n + \frac{\alpha}{2n}(3n^2/2 - n) \quad \text{substituting } (n+1)/2 \text{ for } j \\ &\leq n + 3\alpha n/4 \\ &\leq \alpha n \quad \text{for any constant } \alpha \geq 4 \end{aligned}$$

#### 13.7.0.8 Comments on analyzing the recurrence

(A) Algebra looks messy but intuition suggest that the median is the hardest case and hence can plug $j = n/2$ to simplify without calculus

(B) Analyzing recurrences comes with practice and after a while one can see things more intuitively

**John Von Neumann**:

*Young man, in mathematics you don't understand things. You just get used to them.*