Randomized Algorithms: Element Distinctness

CSE21 Winter 2017, Day 25 (B00), Day 17-18 (A00)

March 15, 2017

http://vlsicad.ucsd.edu/courses/cse21-w17
Given list of positive integers $a_1, a_2, \ldots, a_n$ decide whether all the numbers are distinct or whether there is a repetition, i.e. two positions $i, j$ with $1 \leq i < j \leq n$ such that $a_i = a_j$.

*What algorithm would you choose in general?*
Element Distinctness: HOW

Given list of positive integers \(a_1, a_2, \ldots, a_n\) decide whether all the numbers are distinct or whether there is a repetition, i.e. two positions \(i, j\) with \(1 \leq i < j \leq n\) such that \(a_i = a_j\).

*What algorithm would you choose in general? Can sorting help?*

Algorithm: first sort list and then step through to find duplicates. What's its runtime?

A. \(\Theta(1)\)
B. \(\Theta(n)\)
C. \(\Theta(n \log n)\)
D. \(\Theta(n^2)\)
E. None of the above
Element Distinctness: HOW

Given list of positive integers $a_1, a_2, ..., a_n$ decide whether all the numbers are distinct or whether there is a repetition, i.e. two positions $i, j$ with $1 \leq i < j \leq n$ such that $a_i = a_j$.

*What algorithm would you choose in general? Can sorting help?*

Algorithm: first sort list and then step through to find duplicates. How much memory does it require?
A. $\Theta(1)$  
B. $\Theta(n)$  
C. $\Theta(n \log n)$  
D. $\Theta(n^2)$  
E. None of the above
Element Distinctness: HOW

Given list of positive integers $a_1, a_2, \ldots, a_n$ decide whether all the numbers are distinct or whether there is a repetition, i.e. two positions $i, j$ with $1 \leq i < j \leq n$ such that $a_i = a_j$.

What algorithm would you choose in general? What if we had unlimited memory?
Given list of positive integers \( A = a_1, a_2, \ldots, a_n \),

\textbf{UnlimitedMemoryDistinctness}(A)
1. For \( i = 1 \) to \( n \),
2. If \( M[a_i] = 1 \) then return "Found repeat"
3. Else \( M[a_i] := 1 \)
4. Return "Distinct elements"

What's the runtime of this algorithm?
A. \( \Theta(1) \)
B. \( \Theta(n) \)
C. \( \Theta(n \log n) \)
D. \( \Theta(n^2) \)
E. None of the above
Element Distinctness: HOW

Given list of positive integers $A = a_1, a_2, ..., a_n$,

$\text{UnlimitedMemoryDistinctness}(A)$
1. For $i = 1$ to $n$,
2. If $M[a_i] = 1$ then return "Found repeat"
3. Else $M[a_i] := 1$
4. Return "Distinct elements"

What's the runtime of this algorithm?
A. $\Theta(1)$
B. $\Theta(n)$
C. $\Theta(n \log n)$
D. $\Theta(n^2)$
E. None of the above

What's the memory use of this algorithm?
A. $\Theta(1)$
B. $\Theta(n)$
C. $\Theta(n \log n)$
D. $\Theta(n^2)$
E. None of the above
Element Distinctness: HOW

To simulate having more memory locations: use **Virtual Memory**.

Define **hash function**

\[ h: \{ \text{desired memory locations} \} \rightarrow \{ \text{actual memory locations} \} \]

- Typically we want more memory than we have, so \( h \) is **not one-to-one**.
- How to implement \( h \)?
  - CSE 12, CSE 100.
- Here, let's use hash functions in an algorithm for Element Distinctness.
Element Distinctness: HOW

Given list of positive integers $A = a_1, a_2, \ldots, a_n$, and $m$ memory locations available

HashDistinctness($A, m$)
1. Initialize array $M[1,..,m]$ to all 0s.
2. Pick a hash function $h$ from all positive integers to $1,..,m$.
3. For $i = 1$ to $n$,
4. If $M[h(a_i)] = 1$ then return "Found repeat"
5. Else $M[h(a_i)] := 1$
6. Return "Distinct elements"
Given list of positive integers \( A = a_1, a_2, \ldots, a_n \), and \( m \) memory locations available

**HashDistinctness(\( A, m \))**
1. Initialize array \( M[1,\ldots,m] \) to all 0s.
2. Pick a hash function \( h \) from all positive integers to \( 1,\ldots,m \).
3. For \( i = 1 \) to \( n \),
4. If \( M[ h(a_i) ] = 1 \) then return "Found repeat"
5. Else \( M[ h(a_i) ] := 1 \)
6. Return "Distinct elements"
Element Distinctness: HOW

Given list of positive integers \( A = a_1, a_2, \ldots, a_n \), and \( m \) memory locations available

\[
\text{HashDistinctness}(A, m)
\]
1. Initialize array \( M[1,\ldots,m] \) to all 0s.
2. Pick a hash function \( h \) from all positive integers to 1,\ldots,\( m \).
3. For \( i = 1 \) to \( n \),
4. If \( M[h(a_i)] = 1 \) then return "Found repeat"
5. Else \( M[h(a_i)] := 1 \)
6. Return "Distinct elements"

What's the memory use of this algorithm?
A. \( \Theta(1) \)
B. \( \Theta(n) \)
C. \( \Theta(n \log n) \)
D. \( \Theta(n^2) \)
E. None of the above
Given list of positive integers $A = a_1, a_2, \ldots, a_n$, and $m$ memory locations available

**HashDistinctness($A$, $m$)**

1. Initialize array $M[1,\ldots,m]$ to all 0s.
2. Pick a hash function $h$ from all positive integers to $1,\ldots,m$.
3. For $i = 1$ to $n$,
4. If $M[h(a_i)] = 1$ then return "Found repeat"
5. Else $M[h(a_i)] := 1$
6. Return "Distinct elements"

*But this algorithm might make a mistake!!! When?*
Element Distinctness: WHY

Given list of positive integers $A = a_1, a_2, \ldots, a_n$, and $m$ memory locations available

$\text{HashDistinctness}(A, m)$
1. Initialize array $M[1,..,m]$ to all 0s.
2. Pick a hash function $h$ from all positive integers to $1,..,m$.
3. For $i = 1$ to $n$,
4. If $M[ h(a_i) ] = 1$ then return "Found repeat"
5. Else $M[ h(a_i) ] := 1$
6. Return "Distinct elements"

**Correctness:** 
*Goal is*
If there is a repetition, algorithm finds it
If there is no repetition, algorithm reports "Distinct elements"
Element Distinctness: WHY

Given list of positive integers $A = a_1, a_2, \ldots, a_n$, and $m$ memory locations available

HashDistinctness($A$, $m$)
1. Initialize array $M[1,..,m]$ to all 0s.
2. Pick a hash function $h$ from all positive integers to 1,..,m.
3. For $i = 1$ to $n$,
4. If $M[h(a_i)] = 1$ then return "Found repeat"
5. Else $M[h(a_i)] := 1$
6. Return "Distinct elements"

Correctness: Goal is
If there is a repetition, algorithm finds it ✓
If there is no repetition, algorithm reports "Distinct elements" ❌  Hash Collisions
Resolving collisions with chaining

Hash Table

Each memory location holds a pointer to a linked list, initially empty.

Each linked list records the items that map to that memory location.

Collision means there is more than one item in this linked list
Element Distinctness: HOW

Given list of positive integers \(A = a_1, a_2, \ldots, a_n\), and \(m\) memory locations available

\textbf{ChainHashDistinctness}(A, m)

1. Initialize array \(M[1,\ldots,m]\) to null lists.
2. Pick a hash function \(h\) from all positive integers to \(1,\ldots,m\).
3. For \(i = 1\) to \(n\),
   4. For each element \(j\) in \(M[h(a_i)]\),
   5. If \(a_j = a_i\) then return "Found repeat"
   6. Append \(a_i\) to the tail of the list \(M[h(a_i)]\)
7. Return "Distinct elements"
Element Distinctness: WHY

Given list of positive integers $A = a_1, a_2, \ldots, a_n$, and $m$ memory locations available

ChainHashDistinctness($A$, $m$)
1. Initialize array $M[1,\ldots,m]$ to null lists.
2. Pick a hash function $h$ from all positive integers to $1,\ldots,m$.
3. For $i = 1$ to $n$, 
   4. For each element $j$ in $M[h(a_i)]$,
      5. If $a_j = a_i$ then return "Found repeat"
      6. Append $a_i$ to the tail of the list $M[h(a_i)]$
4. Return "Distinct elements"

Correctness: Goal is
If there is a repetition, algorithm finds it
If there is no repetition, algorithm reports "Distinct elements"
Element Distinctness: MEMORY

Given list of positive integers $A = a_1, a_2, \ldots, a_n$, and $m$ memory locations available

**ChainHashDistinctness**($A$, $m$)
1. Initialize array $M[1,\ldots,m]$ to null lists.
2. Pick a hash function $h$ from all positive integers to $1,\ldots,m$.
3. For $i = 1$ to $n$,
4.   For each element $j$ in $M[h(a_i)]$,
5.     If $a_j = a_i$ then return "Found repeat"
6.   Append $a_i$ to the tail of the list $M[h(a_i)]$
7. Return "Distinct elements"

*What's the memory use of this algorithm?*
Given list of positive integers \( A = a_1, a_2, \ldots, a_n \), and \( m \) memory locations available

**ChainHashDistinctness**\((A, m)\)

1. Initialize array \( M[1,\ldots,m] \) to null lists.
2. Pick a hash function \( h \) from all positive integers to \( 1,\ldots,m \).
3. For \( i = 1 \) to \( n \),
4.  For each element \( j \) in \( M[ h(a_i) ] \),
5.    If \( a_j = a_i \) then return "Found repeat"
6.  Append \( a_i \) to the tail of the list \( M[ h(a_i) ] \)
7. Return "Distinct elements"

*What's the memory use of this algorithm?*

Size of \( M \): \( O(m) \). Total size of all the linked lists: \( O(n) \). Total memory: \( O(m+n) \).
Element Distinctness: WHEN

\begin{algorithm}
\textbf{ChainHashDistinctness}(A, m)
\begin{enumerate}
\item Initialize array $M[1,..,m]$ to null lists.
\item Pick a hash function $h$ from all positive integers to $1,..,m$.
\item For $i = 1$ to $n$,
\item \hspace{1em} For each element $j$ in $M[ h(a_i) ]$,
\item \hspace{2em} If $a_j = a_i$ then return "Found repeat"
\item \hspace{1em} Append $a_i$ to the tail of the list $M[ h(a_i) ]$
\item Return "Distinct elements" \hspace{1em} $\Theta(1)$
\end{enumerate}
\end{algorithm}
Element Distinctness: WHEN

ChainHashDistinctness(A, m)
1. Initialize array M[1,..,m] to null lists.
2. Pick a hash function $h$ from all positive integers to 1,..,m.
3. For i = 1 to n,
4.   For each element j in M[ h(a_i) ],
5.     If $a_j = a_i$ then return "Found repeat"
6.   Append $a_i$ to the tail of the list M[ h(a_i) ]
7. Return "Distinct elements"

Worst case is when we don't find $a_i$: $O( 1 + \text{size of list } M[ h(a_i) ] )$
Element Distinctness: WHEN

ChainHashDistinctness(A, m)
1. Initialize array M[1,..,m] to null lists.
2. Pick a hash function $h$ from all positive integers to 1,..,m.
3. For $i = 1$ to $n$,
4. For each element $j$ in M[ $h(a_i)$ ],
   If $a_j = a_i$ then return "Found repeat"
5. Append $a_i$ to the tail of the list M[ $h(a_i)$ ]
6. Return "Distinct elements"

Worst case is when we don't find $a_i$:
$O(1 + \text{size of list } M[ h(a_i) ] )$
= $O(1 + \# \text{j<i with } h(a_j)=h(a_i) )$
**Element Distinctness: WHEN**

**ChainHashDistinctness(A, m)**
1. Initialize array M[1,..,m] to null lists.
2. Pick a hash function \( h \) from all positive integers to 1,..,m.
3. For \( i = 1 \) to \( n \),
4. For each element \( j \) in M[ \( h(a_i) \) ],
5. If \( a_j = a_i \) then return "Found repeat"
6. Append \( a_i \) to the tail of the list M[ \( h(a_i) \) ]
7. Return "Distinct elements"

**Total time:** \( O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i) \)

\[ = O(n + \text{total # collisions}) \]
Element Distinctness: WHEN

Collisions depend on choice of **hash function**

\[ h: \{ \text{desired memory locations} \} \rightarrow \{ \text{actual memory locations} \} \]

**Ideal hash function model:** each output in \{1,2,\ldots,m\} is equally likely.

So \( h \) is a function that chooses a random number in \{1,2,\ldots,m\} for each input \( a_i \).
Element Distinctness: WHEN

**Total time:** $O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i)$

$= O(n + \text{total # collisions})$

How can I find the total number of collisions?

Doesn’t it depend on the random assignment to memory locations?
Element Distinctness: WHEN

Total time: $O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i)$

= $O(n + \text{total # collisions})$
Element Distinctness: WHEN

Total time: $O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i) \leq O(n + \text{total } \# \text{collisions})$

Expected value!
Element Distinctness: WHEN

Total time: $O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i)$

$= O(n + \text{total } \# \text{collisions})$

LINEARITY OF EXPECTATION!
Element Distinctness: WHEN

**Total time:** \( O(n + \sum_{i=1}^{n} \# \text{ collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i ) \)

\[ = O(n + \text{total # collisions}) \]

*What's the expected total number of collisions?*
Element Distinctness: WHEN

**Total time:** \( O(n + \sum_{i=1}^{n} \text{# collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i ) \)

\[ = O(n + \text{total # collisions}) \]

**What's the expected total number of collisions?**

For each pair \((i,j)\) with \(j<i\), define:

\[ X_{i,j} = 1 \text{ if } h(a_i)=h(a_j) \text{ and } X_{i,j}=0 \text{ otherwise.} \]

Total # of collisions = \( \sum_{(i,j):j<i} X_{i,j} \)
Element Distinctness: WHEN

Total time: \( O(n + \sum_{i=1}^{n} \text{# collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i) \)

\[ = O(n + \text{total # collisions}) \]

What's the expected total number of collisions?

For each pair \((i,j)\) with \(j<i\), define:

\[ X_{i,j} = 1 \text{ if } h(a_i) = h(a_j) \text{ and } X_{i,j} = 0 \text{ otherwise.} \]

Total # of collisions = \( \sum_{(i,j): j<i} X_{i,j} \)

So by linearity of expectation: \( E(\text{total # of collisions}) = \sum_{(i,j): j<i} E(X_{i,j}) \)
Element Distinctness: WHEN

Total time: \( O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i ) \)

\[ = \ O(n + \text{total # collisions}) \]

What's the expected total number of collisions?

For each pair (i,j) with j<i, define:

\[ X_{i,j} = 1 \text{ if } h(a_i) = h(a_j) \text{ and } X_{i,j} = 0 \text{ otherwise.} \]

Total # of collisions = \( \sum_{(i,j):j<i} X_{i,j} \)

What's \( E(X_{i,j}) \)?

A. 1/n
B. 1/m
C. 1/n^2
D. 1/m^2
E. None of the above.
Element Distinctness: WHEN

Total time: $O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i )$

= $O(n + \text{total # collisions})$

What's the expected total number of collisions?

For each pair $(i,j)$ with $j<i$, define:

$X_{i,j} = 1$ if $h(a_i)=h(a_j)$ and $X_{i,j}=0$ otherwise.

Total # of collisions = $\sum_{(i,j):j<i} X_{i,j}$

How many terms are in the sum? That is, how many pairs $(i,j)$ with $j<i$ are there?

A. $n$
B. $n^2$
C. $C(n,2)$
D. $n(n-1)$
Element Distinctness: WHEN

**Total time:** $O(n + \sum_{i=1}^{n} \# \text{collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i )$

$= O(n + \text{total # collisions})$

What's the expected total number of collisions?

For each pair $(i,j)$ with $j<i$, define: $X_{i,j} = 1$ if $h(a_i)=h(a_j)$ and $X_{i,j}=0$ otherwise.

So by linearity of expectation:

$$E(\text{total # of collisions}) = \sum_{(i,j): j<i} E(X_{i,j}) = \binom{n}{2} \frac{1}{m} = O(n^2/m)$$
Element Distinctness: WHEN

**Total time:** $O(n + \sum_{i=1}^{n} \text{# collisions between pairs } a_i \text{ and } a_j, \text{ where } j<i)$

$= O(n + \text{total # collisions})$

**Total expected time:** $O(n + n^2/m)$

In ideal hash model, as long as $m>n$ the total expected time is $O(n)$.

**Note:** This is much better than our original approach using sorting.
Announcements

Final Exam
A00: Wednesday 3/22 7pm
B00: Friday 3/24 11:30am

HW 8
Due Wednesday 11:59pm

OHs, 1-1s
Now is the time!

Two Final Exam Review Sessions
Saturday 3/18 1-3pm
Sunday 3/19 1-3pm
Locations TBD
Final Exam PPs posted
TTK has been started