Dynamic Arrays and Amortized ×

coursera.org/learn/data-structures/exam/Dmth1/dynamic-arrays-and-amortized-analysis/attempt?redirectToCover=true

Dynamic Arrays and Amortized Analysis
Graded Quiz · 30 min

Due Aug 3, 12:29 PM IST

✓ Congratulations! You passed!
TO PASS: 75% or higher

Keep Learning

Retake the assignment in 7h 37m

GRADE
100%

Dynamic Arrays and Amortized Analysis

LATEST SUBMISSION GRADE
100%

1. Let's imagine we add support to our dynamic array for a new operation PopBack (which removes the last element), and that PopBack never reallocates the associated dynamically-allocated array. Calling PopBack on an empty dynamic array is an error.

1 / 1 point

If we have a sequence of 48 operations on an empty dynamic array: 24 PushBack and 24 PopBack (not necessarily in that order), we clearly end with a size of 0.

What are the minimum and maximum possible final capacities given such a sequence of 48 operations on an empty dynamic array? Assume that PushBack doubles the capacity, if necessary, as in lecture.

- (●) minimum: 1, maximum: 32
- ( ) minimum: 1, maximum: 1
- ( ) minimum: 32, maximum: 32
- ( ) minimum: 1, maximum: 24
- ( ) minimum: 24, maximum: 24

✓ Correct
The minimum is achieved when we alternate with one PushBack followed by one PopBack. The size of the array never exceeds 1, so the capacity also never exceeds 1.

Dynamic Arrays and Amortized ×

coursera.org/learn/data-structures/exam/Dmth1/dynamic-arrays-and-amortized-analysis/attempt?redirectToCover=true

Dynamic Arrays and Amortized Analysis
Graded Quiz · 30 min

Due Aug 3, 12:29 PM IST

2. Let's imagine we add support to our dynamic array for a new operation PopBack (which removes the last element). PopBack will reallocate the dynamically-allocated array if the size is $\leq$ the capacity / 2 to a new array of half the capacity. So, for example, if, before a PopBack the size were 5 and the capacity were 8, then after the PopBack, the size would be 4 and the capacity would be 4.

1 / 1 point

Give an example of n operations starting from an empty array that require $O(n^2)$ copies.

- ( ) PushBack $n/2$ elements, and then PopBack $n/2$ elements.
- (●) Let n be a power of 2. Add $n/2$ elements, then alternate $n/4$ times between doing a PushBack of an element and a PopBack.
- ( ) PushBack 2 elements, and then alternate $n/2 - 1$ PushBack and PopBack operations.

✓ Correct
Once we have added $n/2$ elements, the dynamically-allocated array is full (size=$n/2$, capacity=$n/2$). When we add one element, we realoc and copy $n/2$ elements (now: size = $n/2 + 1$, capacity=n) When we then remove an element (with PopBack), we reallocate the dynamically allocated array and copy $n/2$ elements. So, each of the final $n/2$ operations costs $n/2$ copies, for a total of $n^2/4$ moves, or $O(n^2)$.

3. Let's imagine we add support to our dynamic array for a new operation PopBack (which removes the last element). Calling PopBack on an empty dynamic array is an error.

1 / 1 point

PopBack reallocates the dynamically-allocated array to a new array of half the capacity if the size is $\leq$ the capacity / 4 . So, for example, if, before a PopBack the size were 5 and the capacity were 8, then after the PopBack, the size would be 4 and the capacity would be 8. Only after two more PopBack when the size went down to 2 would the capacity go down to 4.

We want to consider the worst-case sequence of any n PushBack and PopBack operations, starting with an empty dynamic array.

What potential function would work best to show an amortized $O(1)$ cost per operation?

- ( ) $\Phi(h) = \max(0, 2 \times \text{size} - \text{capacity})$

empty dynamic array.

What potential function would work best to show an amortized $O(1)$ cost per operation?

- $\Phi(h) = \max(0, 2 \times size - capacity)$
- $\Phi(h) = 2$
- $\Phi(h) = 2 \times size - capacity$
- ◉ $\Phi(h) = \max(2 \times size - capacity, capacity/2 - size)$

✓ Correct
This is a valid potential function since:

- When we start, size=capacity=0, so $\Phi(h_0) = 0$
- $\Phi(h_i) \geq 0$ since, if size > capacity/2, the first term of the max is non-negative, and if size ≤ capacity/2, the second term of the max is non-negative.

The analysis of PushBack remains just as in lecture. The question is what happens when we do a PopBack. Amortized cost = true cost + $\Phi(h_i) - \Phi(h_{i-1})$

- no resize needed: true cost = 1. If size > capacity/2, the change in $\Phi = \Phi(h_i) - \Phi(h_{i-1}) = 2$. If size ≤ capacity/2, the change in $\Phi = 1$. Max total amortized cost is 3.
- Resize needed: true cost = capacity/4 + 1. $\Phi(h_i) = 0, \Phi(h_{i-1})$ = capacity/2 - (capacity/4 + 1) = capacity/4 - 1. Total amortized cost = capacity/4 + 1 - (capacity/4 - 1) = 2.

4. Imagine a stack with a new operation: PopMany which takes a parameter, i, that specifies how many elements to pop from the stack. The cost of this operation is i, the number of elements that need to be popped. 1 / 1 point

Without this new operation, the amortized cost of any operation in a sequence of stack operations (Push, Pop, Top) is $O(1)$ since the true cost of each operation is $O(1)$.

Without this new operation, the amortized cost of any operation in a sequence of stack operations (Push, Pop, Top) is $O(1)$ since the true cost of each operation is $O(1)$.

What is the amortized cost of any operation in a sequence of n stack operations (starting with an empty stack) that includes PopMany (choose the best answers)?

- ☑ $O(1)$ because the sum of the costs of all PopMany operations in a total of n operations is $O(n)$.

✓ Correct
Correct. Since over n operations starting with an empty stack there can be at most n items in the stack, the sum of the costs of the PopMany operations can be at most n. Thus, the total actual costs of n operations is at most $O(n)$, so the amortized cost is $O(n)/n = O(1)$.

- ☑ $O(1)$ because we can place one token on each item in the stack when it is pushed. That token will pay for popping it off with a PopMany.

✓ Correct
Correct. Add a token to each element on the stack as it is pushed. Then, on a PopMany, use those tokens to pay for the popping cost of each. Thus, the amortized cost is 2 which is $O(1)$.

- ☑ $O(1)$ because we can define $\Phi(h) = size$.

✓ Correct
Correct.

Push operations will have an amortized cost of 2: 1 for the push, and 1 for the change in Φ.

Pop operation will have an amortized cost of 0: 1 for the pop, and -1 for the change in Φ.

PopMany operations will have an amortized cost of 0: i for the pop, and -i for the change in Φ.

Thus, the worst-case amortized cost is 2, which is $O(1)$.

Priority Queues: Quiz | Coursera

coursera.org/learn/data-structures/exam/gXO8z/priority-queues-quiz/attempt?redirectToCover=true

Priority Queues: Quiz

Graded Quiz · 30 min

Due Aug 10, 12:29 PM IST

Congratulations! You passed!

TO PASS 65% or higher

Keep Learning

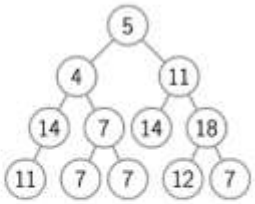
GRADE 100%

Priority Queues: Quiz

LATEST SUBMISSION GRADE

100%

1. 1 / 1 point



How many edges of this binary tree violate the min-heap property? In other words, for how many edges of the tree, the parent value is greater than the value of the child?

4

Correct

Priority Queues: Quiz | Coursera

coursera.org/learn/data-structures/exam/gXO8z/priority-queues-quiz/attempt?redirectToCover=true

Priority Queues: Quiz

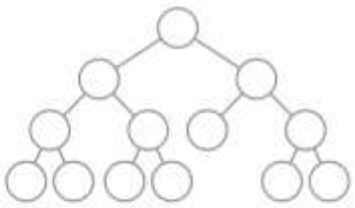
Graded Quiz · 30 min

Due Aug 10, 12:29 PM IST

4

Correct

2. 1 / 1 point



This binary tree contains 13 nodes, and hence we have 13 subtrees here (rooted at each of 13 nodes). How many of them are complete?

11

Correct

3. Consider a complete binary tree represented by an array [19, 14, 28, 15, 16, 7, 27, 15, 21, 21, 5, 2]. 1 / 1 point

How many edges of this tree violate the max-heap property? In other words, for how many edges of the tree,

3. Consider a complete binary tree represented by an array $[19, 14, 28, 15, 16, 7, 27, 15, 21, 21, 3, 2]$. 1 / 1 point

How many edges of this tree violate the max-heap property? In other words, for how many edges of the tree, the parent value is smaller than the value of the child?

5

Correct

4. Assume that a max-heap with 10^9 elements is stored in a complete 5-ary tree. Approximately how many comparisons a call to `Insert()` will make? 1 / 1 point

- 18
- 30
- 8
- 28

Correct
Recall, that to insert a new element, we attach it as a leaf to the last level and let the new node sift up. The number of comparisons required to sift it up is at most the height of the tree. In this case, the height is $\log_5(10^9) \approx 8$.

5. Assume that a max-heap with 10^9 elements is stored in a complete 7-ary tree. Approximately how many comparisons a call to `ExtractMax()` will make? 1 / 1 point

- 50

Priority Queues: Quiz | Coursera
coursera.org/learn/data-structures/exam/gXOBz/priority-queues-quiz/attempt?redirectToCover=true
Priority Queues: Quiz
Graded Quiz · 30 min
Due Aug 10, 12:29 PM IST

Correct
Recall, that to insert a new element, we attach it as a leaf to the last level and let the new node sift up. The number of comparisons required to sift it up is at most the height of the tree. In this case, the height is $\log_5(10^9) \approx 8$.

5. Assume that a max-heap with 10^9 elements is stored in a complete 7-ary tree. Approximately how many comparisons a call to `ExtractMax()` will make? 1 / 1 point

- 50
- 500
- 5

Correct
Recall, that to extract the maximum value, we replace the root node with the last leaf and let this new node sift down. When sifting its down, on each level we need to find the maximum among 7 children. Thus, the worst case running time of `ExtractMax()` in this case is $7 \cdot \log_7(10^9) \approx 50$.

6. Assume that we represent a complete d-ary tree in an array $A[1 \ldots n]$ (this is a 1-based array of size n). What is the right formula for the indices of children of a node number i? 1 / 1 point

- $\{id + 2, \ldots, \min\{n, id + d + 1\}\}$
- $\{(i-1)d + 1, \ldots, \min\{n, (i-1)d + d\}\}$
- $\{(i-1)d + 2, \ldots, (i-1)d + d + 1\}$
- $\{(i-1)d + 2, \ldots, \min\{n, (i-1)d + d + 1\}\}$

Correct

Quiz: Disjoint Sets | Coursera

coursera.org/learn/data-structures/exam/mOsJP/quiz-disjoint-sets/attempt?redirectToCover=true

Quiz: Disjoint Sets

Graded Quiz · 30 min

Due Aug 10, 12:29 PM IST

Congratulations! You passed!

TO PASS 50% or higher

Keep Learning

GRADE 100%

Quiz: Disjoint Sets

LATEST SUBMISSION GRADE

100%

1. Consider the following program:

1 point

```
for i from 1 to 10:
  MakeSet(i)
Union(1, 10)
Union(7, 3)
Union(3, 2)
Union(5, 4)
Union(7, 11)
Union(7, 8)
Union(7, 9)
Union(10, 7)
Union(9, 6)
print(Find(2))
print(Find(3))
print(Find(10))
print(Find(9))
```

Assume that the disjoint sets data structure is implemented as an array `smallest[1...12]`: `smallest[i]` is equal to the smallest element in the set containing i.

What is the output of the following program? As an answer, enter four integers separated by spaces.

1 3 3 1

Quiz: Disjoint Sets | Coursera

coursera.org/learn/data-structures/exam/mOsJP/quiz-disjoint-sets/attempt?redirectToCover=true

Quiz: Disjoint Sets

Graded Quiz · 30 min

Due Aug 10, 12:29 PM IST

2. Consider the program:

1 point

```
for i from 1 to 12:
  MakeSet(i)
Union(7, 10)
Union(7, 3)
Union(3, 1)
Union(5, 4)
Union(7, 11)
Union(7, 8)
Union(7, 9)
Union(12, 3)
Union(6, 4)
```

Assume that the disjoint sets data structure is implemented as disjoint trees with union by rank heuristic.

Compute the product of the heights of the resulting trees after executing the code. For example, for a forest consisting of four trees of height 1, 2, 3, 1 the answer would be 6. (Recall that the height of a tree is the number of edges on a longest path from the root to a leaf. In particular, the height of a tree consisting of just one node is equal to 0.)

2

✓ Correct

Right. There will be 3 trees of height 1, 1, and 2.

5. Consider the following program:

1 point

```
for i from 1 to n
  MakeSet(i)
for i from 1 to n-1:
  Union(i, i+1)
```

Quiz: Disjoint Sets | Coursera

coursera.org/learn/data-structures/exam/mOsJP/quiz-disjoint-sets/attempt?redirectToCover=true

Quiz: Disjoint Sets

Graded Quiz · 30 min

Due Aug 10, 12:29 PM IST

3. Consider the following program:

1 point

```
1 for i from 1 to n:
2   MakeSet(i)
3 for i from 1 to n-1:
4   Union(i, i+1)
```

Assume that the disjoint sets data structure is implemented as disjoint trees with union by rank heuristic.

What is the number of trees in the forest and the maximum height of a tree in this forest after executing this code? (Recall that the height of a tree is the number of edges on a longest path from the root to a leaf. In particular, the height of a tree consisting of just one node is equal to 0.)

- $\log_2 n$ trees, the maximum height is 1.
- Two trees, both of height 1.
- One tree of height $\log_2 n$.
- n trees, the maximum height is 1.
- **One tree of height 1.** ✓
- n/2 trees, the maximum height is 2.

✓ Correct

4. Consider the following program:

1 point

```
1 for i from 1 to n:
2   MakeSet(i)
3 for i from 1 to n:
4   Union(i, 2*i)
5 for i from 1 to n:
6   Union(i, 2*i)
```

Quiz: Disjoint Sets | Coursera

coursera.org/learn/data-structures/exam/mOsJP/quiz-disjoint-sets/attempt?redirectToCover=true

Quiz: Disjoint Sets

Graded Quiz · 30 min

Due Aug 10, 12:29 PM IST

- **One tree of height 1.** ✓
- n/2 trees, the maximum height is 2.

✓ Correct

4. Consider the following program:

1 point

```
1 for i from 1 to n:
2   MakeSet(i)
3 for i from 1 to n:
4   Union(i, 2*i)
5 for i from 1 to n:
6   Union(i, 2*i)
7 for i from 1 to n:
8   Union(i, 2*i)
9 for i from 1 to n:
10  Find(i)
11
```

Assume that the disjoint sets data structure is implemented as disjoint trees with union by rank heuristic and with path compression heuristic.

Compute the maximum height of a tree in the resulting forest. (Recall that the height of a tree is the number of edges on a longest path from the root to a leaf. In particular, the height of a tree consisting of just one node is equal to 0.)

1

✓ Correct

There is at least one tree of height 1 in the forest. Also, all trees have height at most 1, since the last for-loop calls Find() for all 60 elements. Since path compression is used, each non-root node will be attached directly to the corresponding root in this loop, and hence all the trees will have height at most 1.

Hashing | Coursera

coursera.org/learn/data-structures/quiz/neIOL/hashing/attempt?redirectToCover=true

Hashing
Practice Quiz • 30 min

Congratulations! You passed!
TO PASS 80% or higher

Keep Learning

GRADE
100%

Hashing

TOTAL POINTS 3

1. What is the minimum size of an array that can be used in the direct addressing scheme to store a map from 7-digit phone numbers to names? 1 / 1 point

- 1000000
- 10000000
- 20000000

Correct
Correct! 7-digit phone numbers correspond to integers from 0 to 9999999.

2. If it is guaranteed that the total length of all occurrences of a $Pattern$ in a $Text$ is at most L, which of the below estimates of the average running time of Rabin-Karp's algorithm to find all occurrences of the $Pattern$ in the $Text$ is the most tight out of the correct ones? 1 / 1 point

- $O(|Text||Pattern|L)$
- $O(|Text| + |Pattern| + L)$
- $O(|Text| + |Pattern|)$

Hashing | Coursera

coursera.org/learn/data-structures/quiz/neIOL/hashing/attempt?redirectToCover=true

Hashing
Practice Quiz • 30 min

Correct
Correct! 7-digit phone numbers correspond to integers from 0 to 9999999.

2. If it is guaranteed that the total length of all occurrences of a $Pattern$ in a $Text$ is at most L, which of the below estimates of the average running time of Rabin-Karp's algorithm to find all occurrences of the $Pattern$ in the $Text$ is the most tight out of the correct ones? 1 / 1 point

- $O(|Text||Pattern|L)$
- $O(|Text| + |Pattern| + L)$
- $O(|Text| + |Pattern|)$
- $O(|Text||Pattern| + L)$

Correct
Correct! Estimate from the lecture is $O(|Text| + (q + 1)|Pattern|)$, where q is the number of occurrences of the $Pattern$ in the $Text$, and $L = q|Pattern|$ in this case.

3. Let us slightly change the polynomial hash function for strings and set $h(S) = (\sum_{i=0}^{|S|-1} x^{|S|-1-i} S[i]) \bmod p$. Let us fix some $Text$ and some $Pattern$. Denote by $H[i]$ the hash function of the substring $Text[i..i + |Pattern| - 1]$ of the $Text$ starting from position i and having the same length as $Pattern$ (for all appropriate positions i where the $Pattern$ can occur in the $Text$). Which of the below formulas is the correct recurrence to compute $H[i + 1]$ given $H[i]$? 1 / 1 point

- $H[i] = (xH[i + 1] + Text[i] - x^{|Pattern|}Text[i + |Pattern|]) \bmod p$
- $H[i + 1] = (xH[i] + Text[i + |Pattern| - 1] - x^{|Pattern|}Text[i]) \bmod p$
- $H[i + 1] = (xH[i] + x^{|Pattern|}Text[i + |Pattern| - 1] - Text[i]) \bmod p$

Hashing | Coursera

coursera.org/learn/data-structures/quiz/neIOL/hashing/attempt?redirectToCover=true

Hashing

Practice Quiz • 30 min

2. If it is guaranteed that the total length of all occurrences of a $Pattern$ in a $Text$ is at most L, which of the below estimates of the average running time of Rabin-Karp's algorithm to find all occurrences of the $Pattern$ in the $Text$ is the most tight out of the correct ones?

1 / 1 point

- $O(|Text||Pattern|L)$
- ◉ $O(|Text| + |Pattern| + L)$
- $O(|Text| + |Pattern|)$
- $O(|Text||Pattern| + L)$

✓ Correct

Correct! Estimate from the lecture is $O(|Text| + (q + 1)|Pattern|)$, where q is the number of occurrences of the $Pattern$ in the $Text$, and $L = q|Pattern|$ in this case.

3. Let us slightly change the polynomial hash function for strings and set $h(S) = (\sum_{j=0}^{|S|-1} x^{|S|-1-j} S[j]) \bmod p$. Let us fix some $Text$ and some $Pattern$. Denote by $H[i]$ the hash function of the substring $Text[i..i + |Pattern| - 1]$ of the $Text$ starting from position i and having the same length as $Pattern$ (for all appropriate positions i where the $Pattern$ can occur in the $Text$). Which of the below formulas is the correct recurrence to compute $H[i + 1]$ given $H[i]$?

1 / 1 point

- $H[i] = (xH[i + 1] + Text[i] - x^{|Pattern|}Text[i + |Pattern|]) \bmod p$
- $H[i + 1] = (xH[i] + Text[i] - |Pattern| - 1] - x^{|Pattern|}Text[i]) \bmod p$
- $H[i + 1] = (xH[i] - x^{|Pattern|}Text[i] + |Pattern|] - Text[i]) \bmod p$
- ◉ $H[i + 1] = (xH[i] - Text[i + |Pattern|] - x^{|Pattern|}Text[i]) \bmod p$

✓ Correct

Correct! When we move one position to the right from position i, each term must increase the power of x in it. In one, the first term $x^{|Pattern|}Text[i]$ must be subtracted after that, and a new term $Text[i + |Pattern|]$

Hash Tables and Hash Functions

coursera.org/learn/data-structures/exam/ACuTU/hash-tables-and-hash-functions/attempt?redirectToCover=true

Hash Tables and Hash Functions

Graded Quiz • 30 min

Due Aug 17, 12:29 PM IST

✓ Congratulations! You passed!

TO PASS 75% or higher

Keep Learning

Retake the assignment in 7h 58m

GRADE 100%

Hash Tables and Hash Functions

LATEST SUBMISSION GRADE

100%

1. What is the size of the array needed to store integer keys with up to 12 digits using direct addressing?

1 / 1 point

- 12
- ◉ 10^{12}
- 9^{12}

✓ Correct

This is the number of all integers with up to 12 digits.

2. What is the maximum possible chain length for a hash function $h(x) = x \bmod 1000$ used with a hash table of size 1000 for a universe of all integers with at most 12 digits?

1 / 1 point

- 10^{12}
- ◉ 10^9
- 1

- 10^{12}
- 1

✓ Correct
When the values of the last 3 digits are fixed, there are 10^9 numbers with at most 12 digits.

3. You want to hash integers from 0 up to 1000000. What can be a good choice of p for the universal family? 1 / 1 point

- 999997
- 1000002
- 1000003

✓ Correct
This is a prime number bigger than 1000000.

4. How can one build a universal family of hash functions for integers between −1000000 (minus one million) and 1000000 (one million)? 1 / 1 point

- First, add 1000000 to each integer. Then use the universal family for integers with $p = 1000003$.
- First, add 1000000 to each integer and get the range of integers between 0 and 2000000. Then use the universal family for integers with $p = 2000003$.
- Take the universal family for integers with $p = 1000003$.

✓ Correct