صف اولویت

سید ناصر رضوی www.snrazavi.ir
۱۳۹۵

فهرست مطالب

- صف اولویت.
  - واسط (API)
  - پیاده‌سازی‌های اولیه
  - هرم‌های دودویی
  - مرتب‌سازی هرمی
  - کاربردها

۳ واسط صف اولویت

صف اولویت

- کلکسیون‌ها. درج و حذف و عناصر.
  - کدام عنصر حذف شود؟

- پشته. حذف عنصری که دیرتر از بقیه اضافه شده است.
- صف. حذف عنصری که زودتر از بقیه اضافه شده است.
- صف تصادفی. حذف یک عنصر به صورت تصادفی.
- صف اولویت. حذف بزرگ‌ترین (یا کوچک‌ترین) عنصر.

عمل	عنصر	مقدار برگشتی
درج	P	
درج	Q	
درج	E	
حذف بزرگترین		Q
درج	X	
درج	A	
درج	M	
حذف بزرگترین		X
درج	P	
درج	L	
درج	E	
حذف بزرگترین		P

واسط صف اولویت

- نیازمندی. عناصر عمومی باید قابل مقایسه باشند.

public class MaxPQ <Key extends Comparable<Key>>	
MaxPQ()	ایجاد یک صف اولویت تهی
MaxPQ(Key[] a)	ایجاد یک صف اولویت با کلیدهای داده شده
void insert(Key v)	درج یک عنصر در صف اولویت
Key delMax()	حذف بزرگ‌ترین کلید و برگرداندن آن
Key max()	برگرداندن بزرگ‌ترین کلید
boolean isEmpty()	بررسی تهی بودن صف اولویت
int size()	برگرداندن تعداد عناصر موجود در صف اولویت

چه اشیایی قابل مقایسه هستند؟

- اشیاء متناظر با انواع داده‌ای اولیه.

Byte, Short, Integer, Long, Float, Double

- کاراکترها و رشته‌ها.

Character, String

- و به طوری کلی هر شی‌ای که واسط **Comparable** را پیاده‌سازی کند.

```
public class MyClass implements Comparable<MyClass> {

    ...

    public int compareTo(MyClass myClass)
    {
        ...
    }
}
```

چه اشیایی قابل مقایسه هستند؟


```
public class Date implements Comparable<Date>
{
   private final int day;
   private final int month;
   private final int year;

   public Date(int d, int m, int y)
   {  day = d; month = m; year = y; }

   public int day()   { return day;   }
   public int month() { return month; }
   public int year()  { return year;  }

   public int compareTo(Date that)
   {
      if (this.year  > that.year ) return +1;
      if (this.year  < that.year ) return -1;
      if (this.month > that.month) return +1;
      if (this.month < that.month) return -1;
      if (this.day   > that.day  ) return +1;
      if (this.day   < that.day  ) return -1;
      return 0;
   }
}
```

چه اشیایی قابل مقایسه هستند؟


```
public class Transaction implements Comparable<Transaction>
{
   private final String who;
   private final Date   when;
   private final double amount;

   public Transaction(String name, Date date, double amnt)
   {  who = name; when = date; amount = amnt; }

   public String who() { return who;    }
   public Date when()  { return when;   }
   public int amount() { return amount; }

   public int compareTo(Transaction that)
   {
      if      (this.amount > that.amount) return +1;
      else if (this.amount < that.amount) return -1;
      else                                return  0;
   }

}
```

برخی از کاربردها

- شبیه‌سازی رویدادگرا [مشتریان در یک صف، تصادم ذرات]
- فشرده‌سازی داده‌ها [کدهای هافمن]
- جستجوی گراف [الگوریتم دیکسترا، الگوریتم پریم]
- هوش مصنوعی [جستجوی **A***]
- آمار [نگهداری **M** مقدار بزرگ‌تر از یک دنباله]
- سیستم‌های عامل [توازن بار، برخورد با وقفه‌ها]
- فیلتر کردن هرزنامه‌ها [فیلتر بیزی]

- صف اولویت. تعمیم صف، پشته و صف تصادفی.

یک مثال از کاربرد صف اولویت

- چالش. یافتن **M** بزرگ‌ترین عنصر در دنباله‌ای از **N** عنصر در حال جریان.
  - تشخیص کلاهبرداری: جدا کردن تراکنش‌های بزرگ
  - نگهداری فایل‌ها: یافتن بزرگ‌ترین فایل‌ها و دایرکتوری‌ها
- محدودیت. حافظه ناکافی برای ذخیره همه **N** عنصر [**N** بسیار بزرگ است]

```
% more tinyBatch.txt
Turing       6/17/1990   644.08
vonNeumann   3/26/2002  4121.85
Dijkstra     8/22/2007  2678.40
vonNeumann   1/11/1999  4409.74
Dijkstra    11/18/1995   837.42
Hoare        5/10/1993  3229.27
vonNeumann   2/12/1994  4732.35
Hoare        8/18/1992  4381.21
Turing       1/11/2002    66.10
Thompson     2/27/2000  4747.08
Turing       2/11/1991  2156.86
Hoare        8/12/2003  1025.70
```

```
% java TopM 5 < tinyBatch.txt
Thompson     2/27/2000  4747.08
vonNeumann   2/12/1994  4732.35
vonNeumann   1/11/1999  4409.74
Hoare        8/18/1992  4381.21
vonNeumann   3/26/2002  4121.85
```

یک مثال از کاربرد صف اولویت

- چالش. یافتن **M** بزرگ‌ترین عنصر در دنباله‌ای از **N** عنصر در حال جریان.

```
MinPQ<Transaction> pq = new MinPQ<Transaction>();

while (StdIn.hasNextLine()) {
   String line = StdIn.readLine();
   Transaction item = new Transaction(line);
   pq.insert(item);
   if (pq.size() > M)
      pq.delMin();
}
```

اشیای تراکنش برحسب مقدار با یکدیگر قابل مقایسه هستند

implementation	time	space
sort	N log N	N
elementary PQ	M N	M
binary heap	**M log N**	**M**
best in theory	N	M

پیاده‌سازی ابتدایی

پیاده‌سازی به وسیله آرایه‌ی مرتب و نامرتب

عمل	عنصر	مقدار برگشتی	اندازه	محتویات (آرایه نامرتب)	محتویات (آرایه مرتب)
درج	P		1	P	P
درج	Q		2	P Q	P Q
درج	E		3	P Q E	E P Q
حذف بزرگ‌ترین		Q	2	P E	E P
درج	X		3	P E X	E P X
درج	A		4	P E X A	A E P X
درج	M		5	P E X A M	A E M P X
حذف بزرگ‌ترین		X	4	P E M A	A E M P
درج	P		5	P E M A P	A E M P P
درج	L		6	P E M A P L	A E L M P P
درج	E		7	P E M A P L E	A E E L M P P
حذف بزرگ‌ترین		P	6	E E M A L E	A E E L M P

صف اولویت: پیاده‌سازی با آرایه نامرتب


```
public class UnorderedMaxPQ<Key extends Comparable<Key>>
{
   private Key[] pq;       // pq[i] = ith element on pq
   private int N;          // number of elements on pq

  public UnorderedMaxPQ<int capacity)
  {  pq = (Key[]) new Comparable[capacity]; }

  public boolean isEmpty()
  {  return N == 0; }

  public void insert(Key x)
  {  pq[N++] = x; }

  public Key delMax()
  {
     int max = 0;
     for (int i = 1; i < N; i++)
        if (less(max, i)) max = i;
     exch(max, N - 1);
     return pq[--N];
  }

}
```

پیاده‌سازی اولیه صف مرتب

- چالش. پیاده‌سازی همه عملیات به صورت کارا.

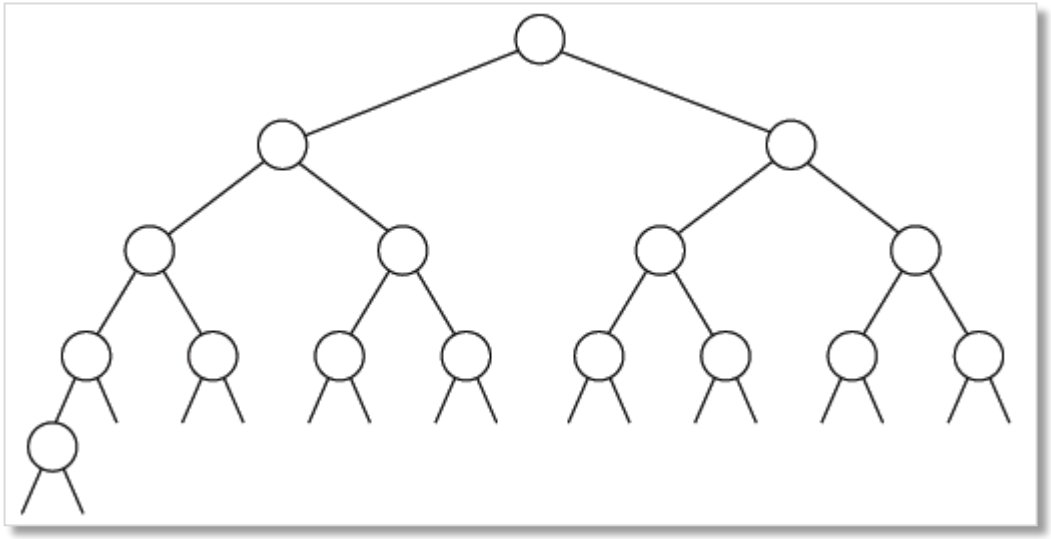
implementation	insert	delMax	max
Unordered array	1	N	N
Ordered array	N	1	1
goal	log N	log N	log N

۱۶ هرم دودویی

درخت دودویی

- درخت دودویی.
  - یا تهی است یا حاوی یک گره با دو پیوند به زیردرخت‌های دودویی چپ و راست است.

- درخت دودویی کامل.
  - بدون در نظر گرفتن آخرین سطح کاملاً متوازن است.
  - برگ‌ها در آخرین سطح از چپ به راست چیده شده‌اند.

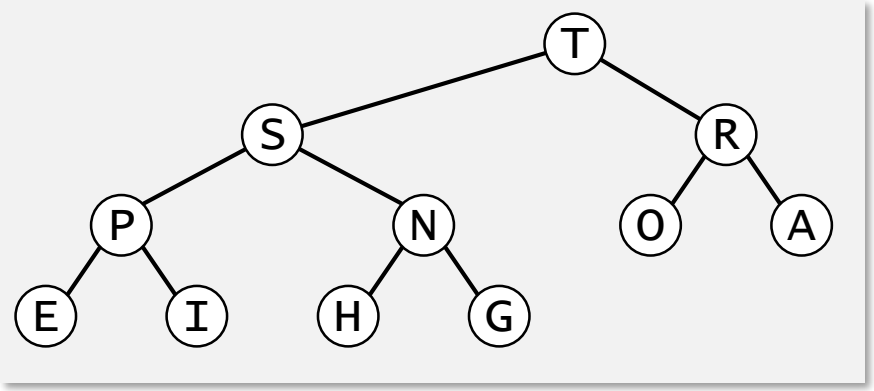


ارتفاع یک درخت دودویی کامل با N گره:
$\lfloor \log N \rfloor$

درخت دودویی کامل در طبیعت

هرم دودویی

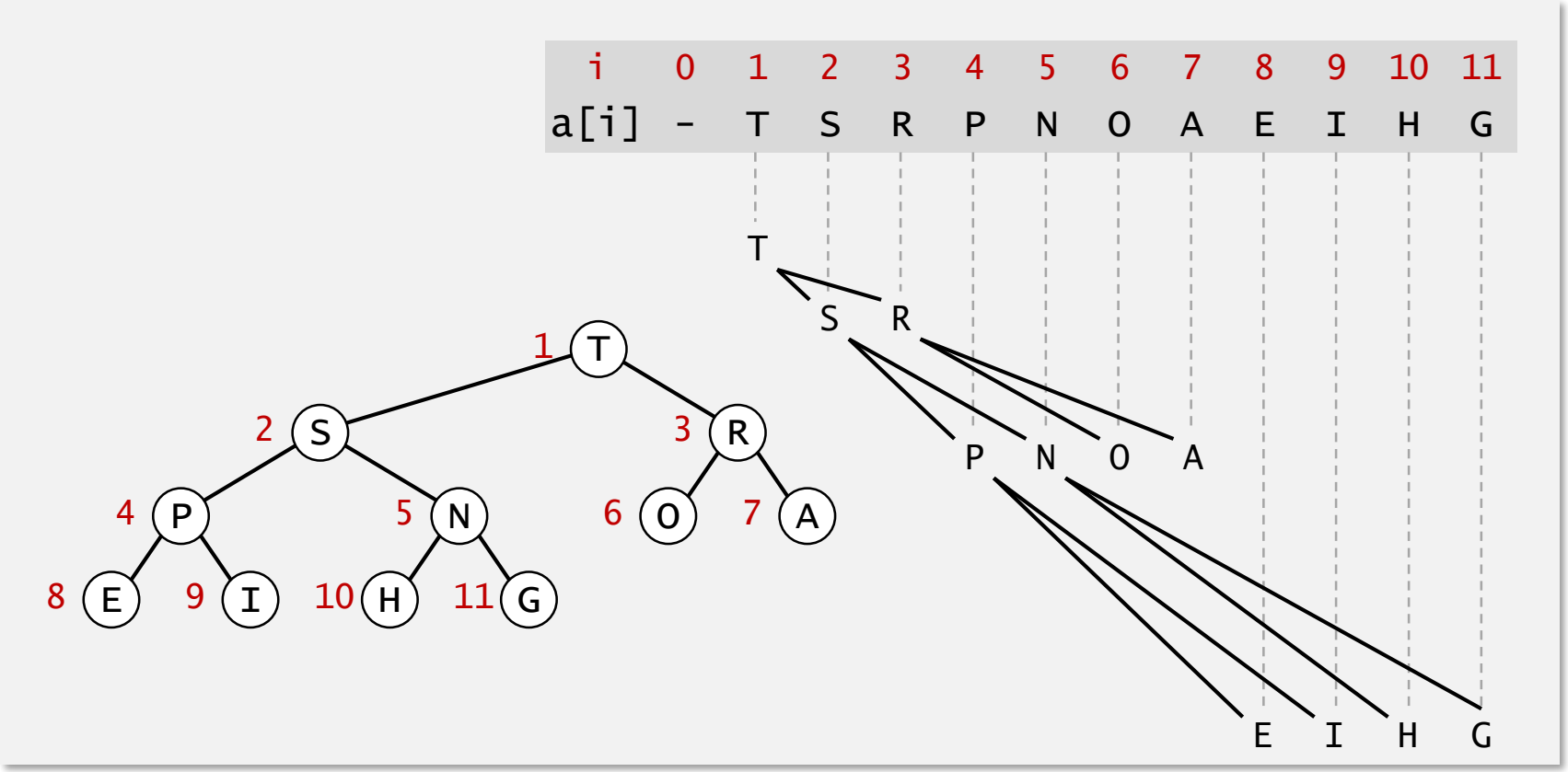
- هرم دودویی. یک درخت دودویی کامل با خاصیت هرمی.



- خاصیت هرمی.
  - هر گره شامل یک کلید است.
  - کلید یک گره هرگز از کلید فرزندانش کوچک‌تر نیست.

پیاده‌سازی با آرایه

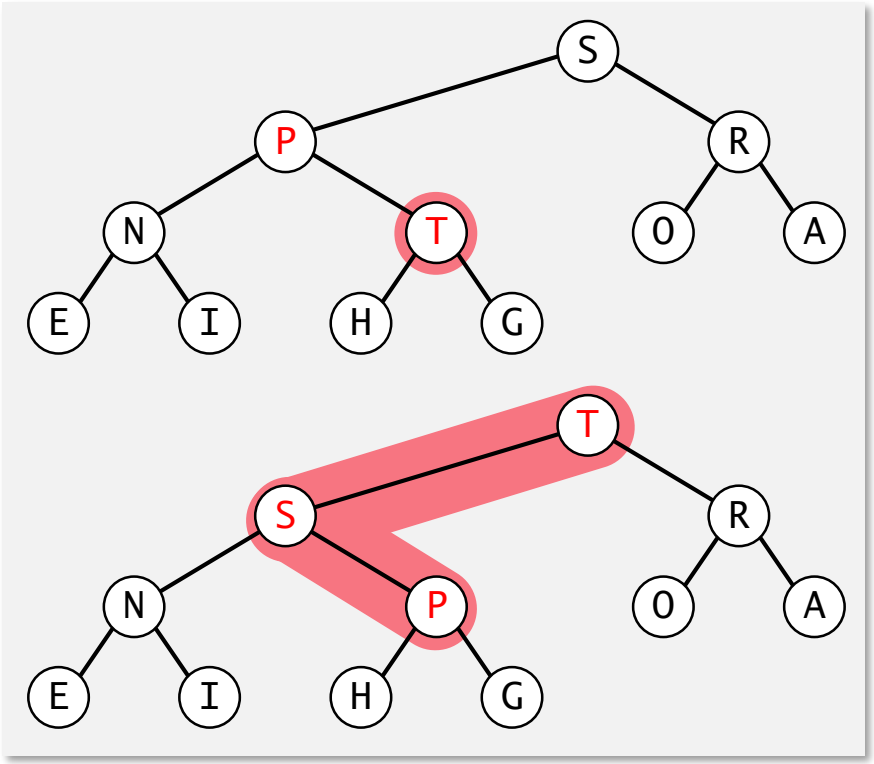
- خواص. بزرگ‌ترین کلید در `a[1]` قرار دارد. به ازای هر گره در مکان **k**:
  - پدر آن گره در مکان **k/2** قرار دارد.
  - فرزندان چپ و راست (در صورت وجود) به ترتیب در مکان‌های **2k** و **2k+1** قرار دارند.



بالا رفتن در درخت هرمی

- سناریو. پس از انجام یک عمل، کلید یک گره از کلید پدرش بزرگ‌تر شده است.

- برقراری مجدد خاصیت هرمی.
  - کلید فرزند را با کلید پدر تعویض کن.
  - این عمل را در صورت لزوم تکرار کن.

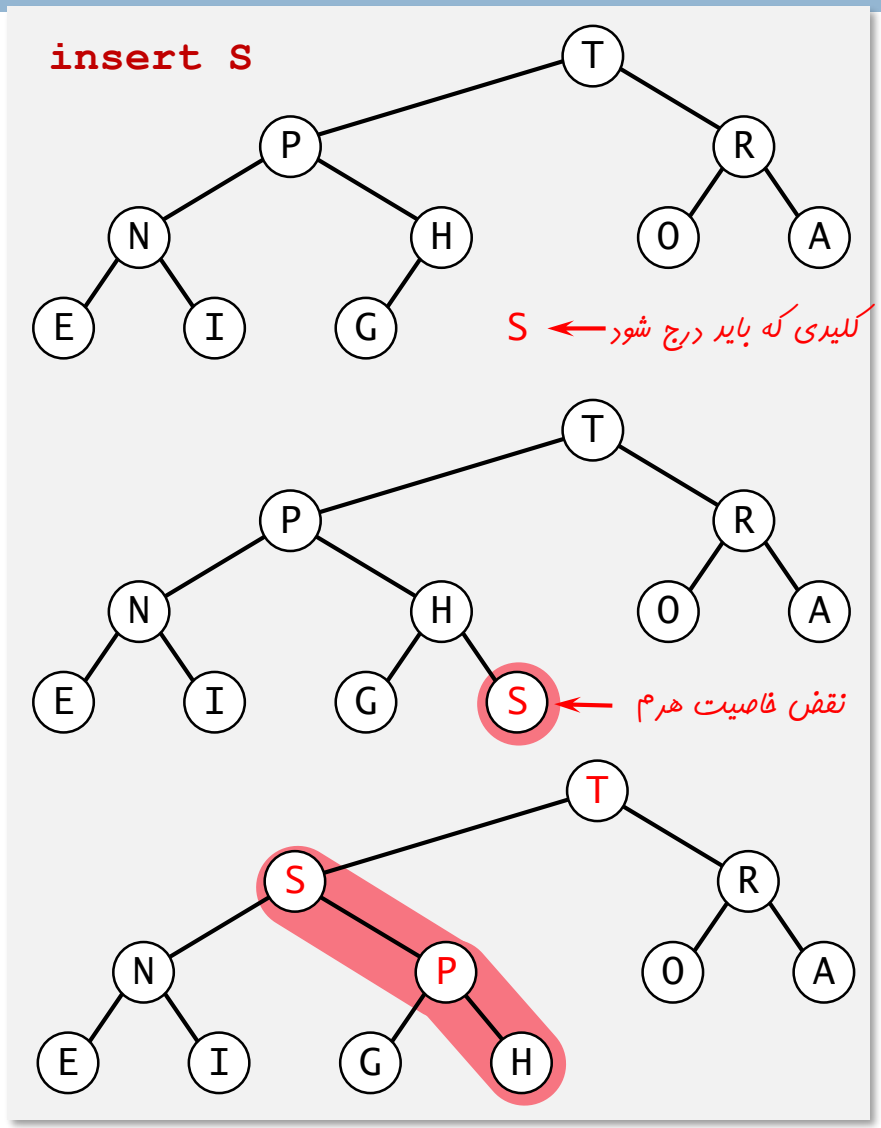



```
private void swim(int k)
{
   while (k > 1 && less(k/2, k))
   {
      ecxh(k, k/2);
      k = k / 2;
   }
}
```

درج در درخت هرمی

- درج.
  - گره را به انتهای درخت اضافه کن،
  - سپس در صورت لزوم آن را بالا ببر.

- هزینه.
  - حداکثر `1 + lg N` مقایسه!

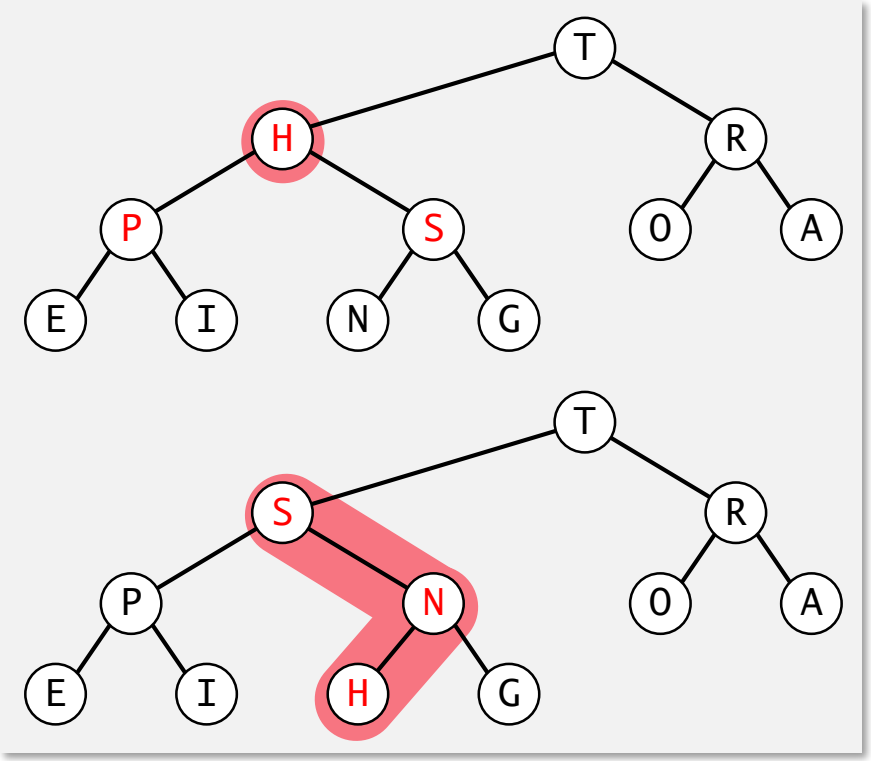



```
public void insert(Key x)
{
   p[++N] = x;
   swim(N);
}
```

پایین رفتن در درخت هرمی

- سناریو. پس از انجام یک عمل، کلید یک گره از کلید حداقل یکی از فرزندانش کوچک‌تر شده است.

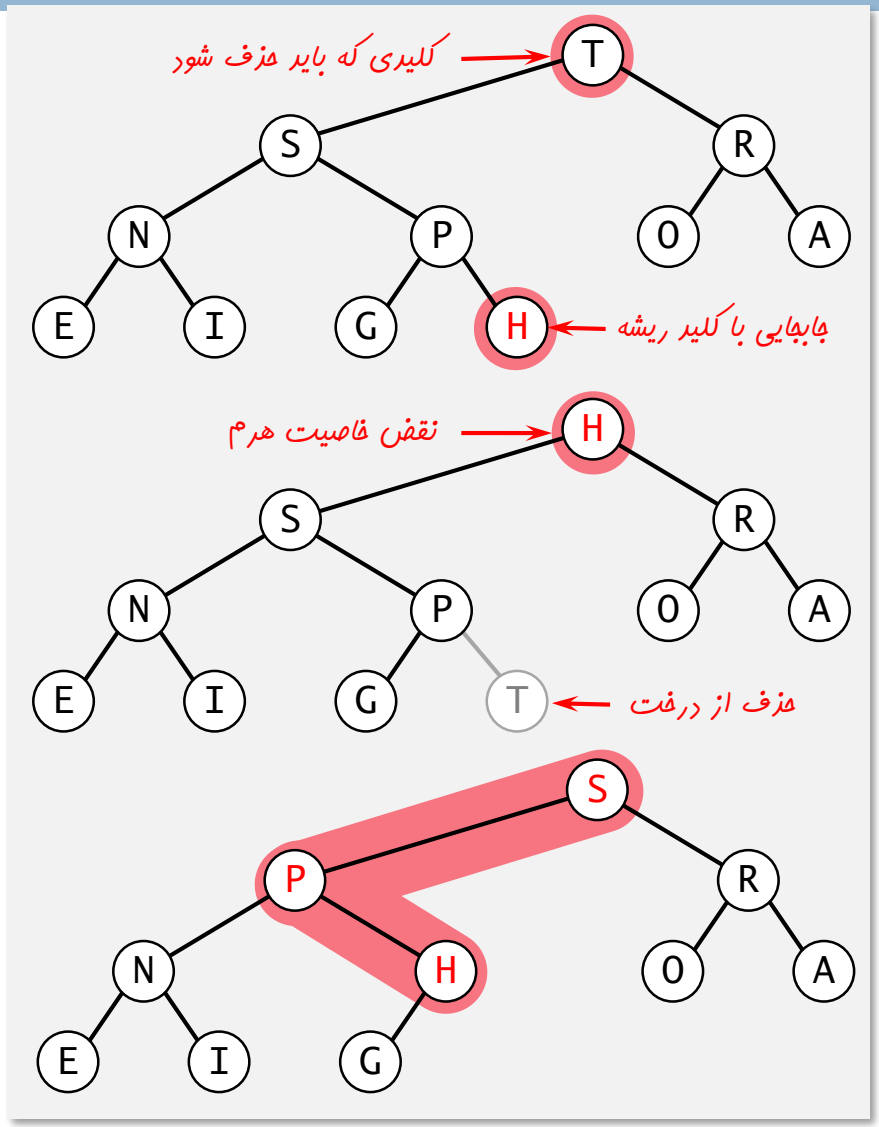
- برقراری مجدد خاصیت هرمی.
  - کلید را با کلید فرزند بزرگ‌تر تعویض کن.
  - این عمل را در صورت لزوم تکرار کن.




```
private void sink(int k) {
   while (2*k <= N)
   {
      int j = 2 * k;
      if (j < N && less(j, j+1)) j++;
      if (!less(k, j)) break;
      ecxh(k, j);
      k = j;
   }
}
```

حذف بزرگ‌ترین عنصر از درخت هرمی

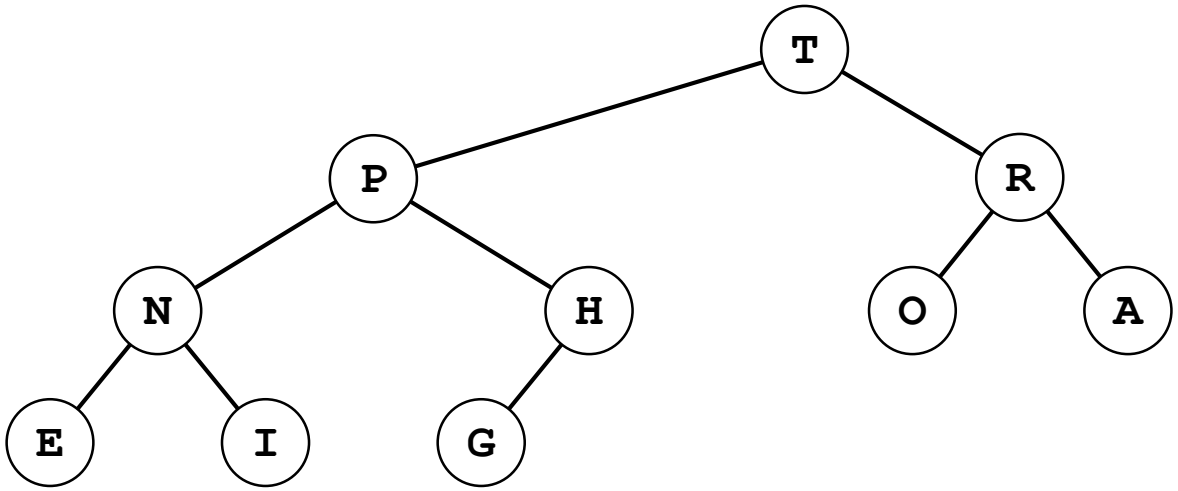
- حذف بزرگ‌ترین عنصر.
  - کلید ریشه را با کلید آخرین گره جابجا کن
  - سپس، کلید ریشه را تا موقعی که نیاز است به سمت پایین حرکت بده




```
public Key delMax()
{
   Key max = pq[1];
   exch(1, N--);
   sink(1);
   pq[N + 1] = null;
   return max;
}
```

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

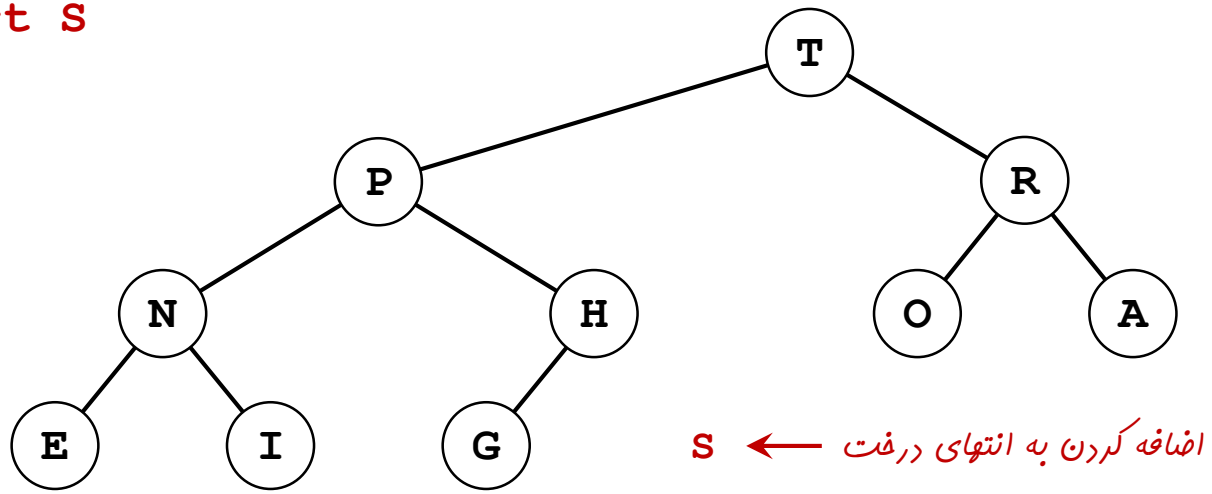


T	P	R	N	H	O	A	E	I	G	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
insert S
```

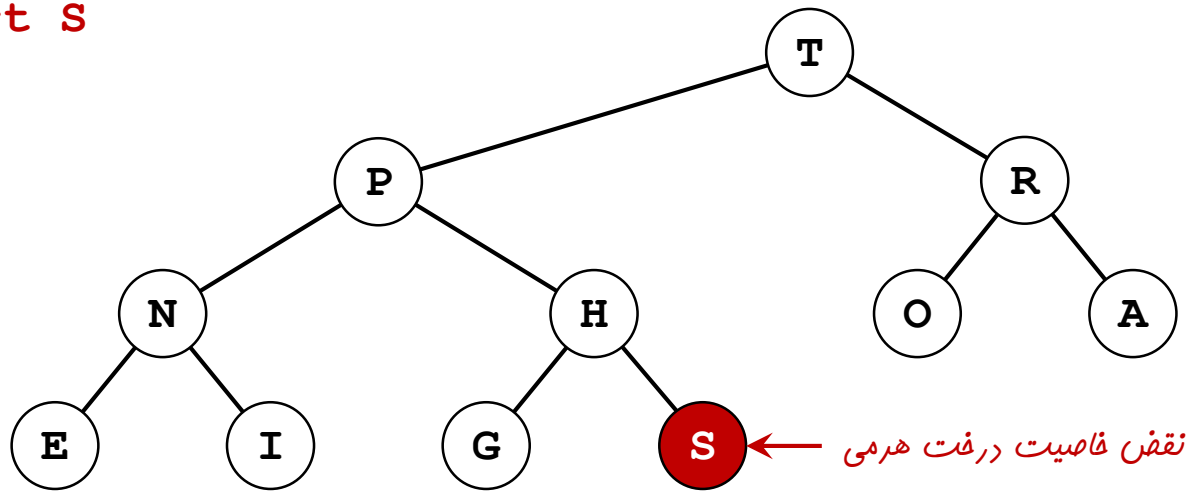


T	P	R	N	H	O	A	E	I	G	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
insert S
```

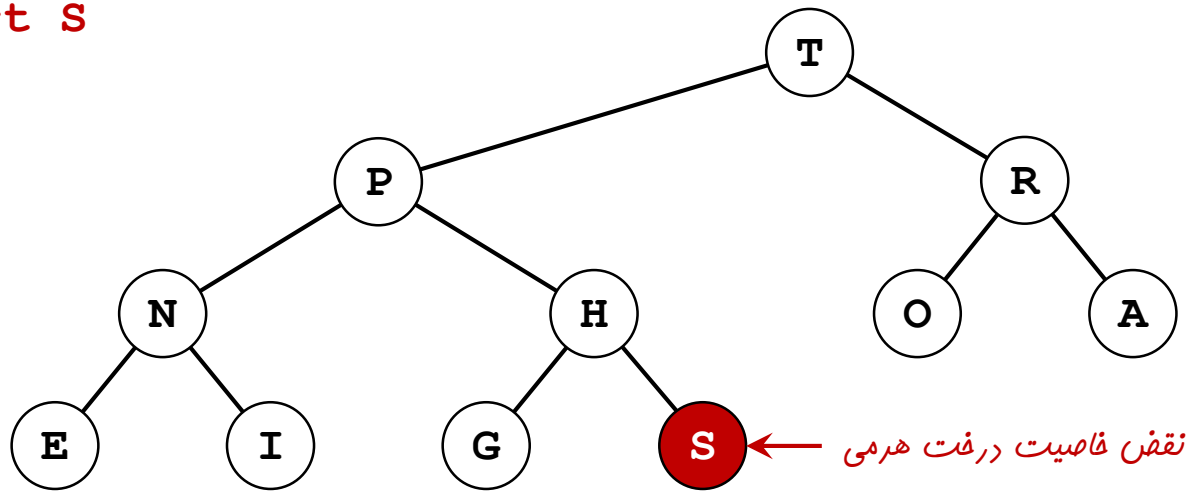


T	P	R	N	H	O	A	E	I	G	S

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
insert S
```

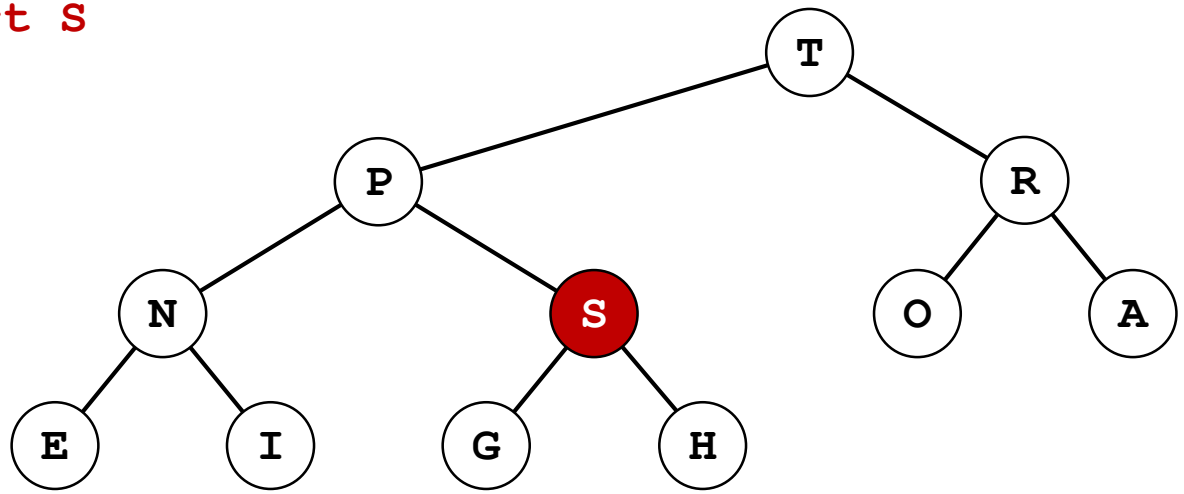


T	P	R	N	H	O	A	E	I	G	S
				5						11

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
insert S
```

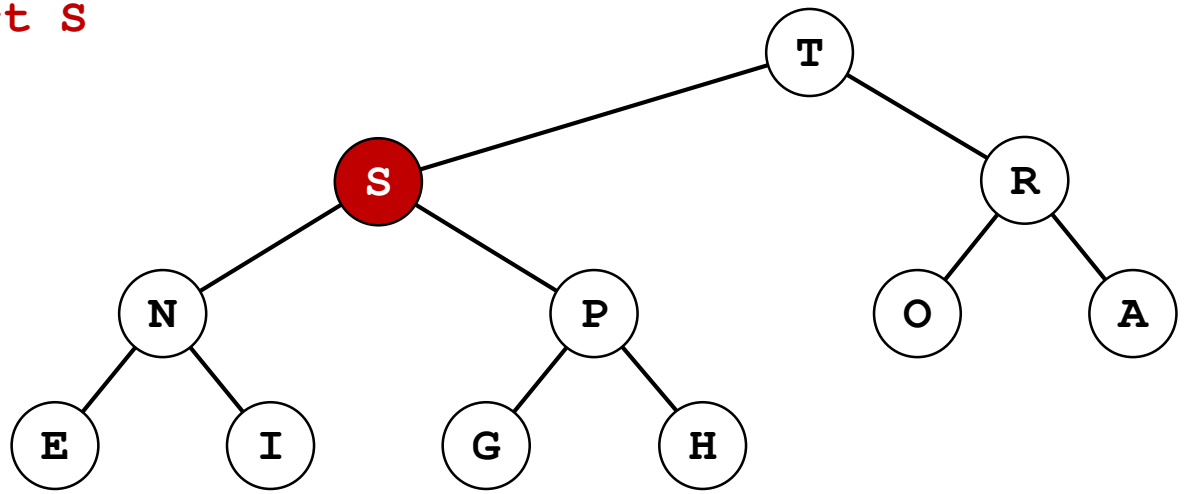


T	P	R	N	S	O	A	E	I	G	H
	2			5						11

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

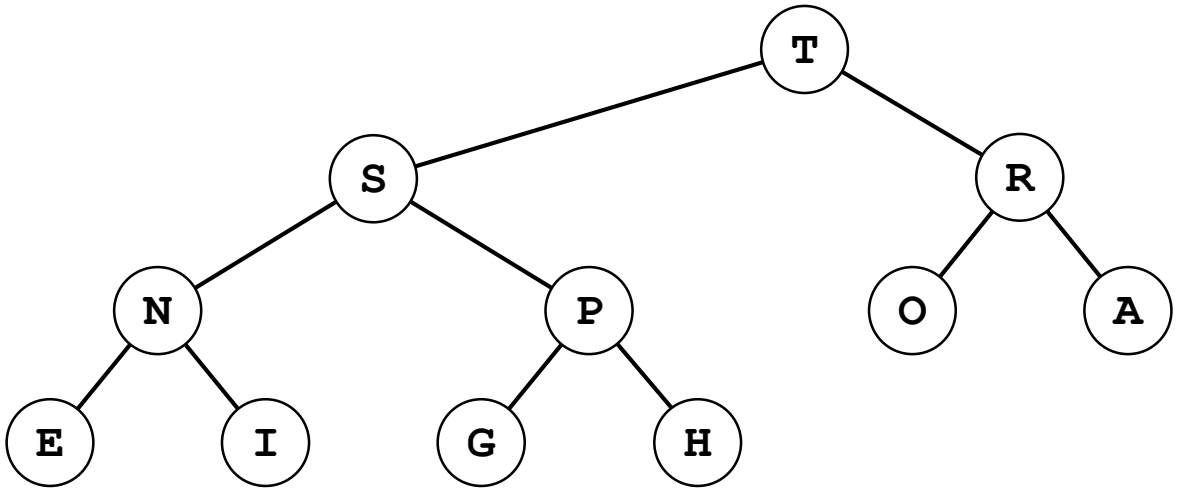
```
insert S
```



T	S	R	N	P	O	A	E	I	G	H
	2			5						

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

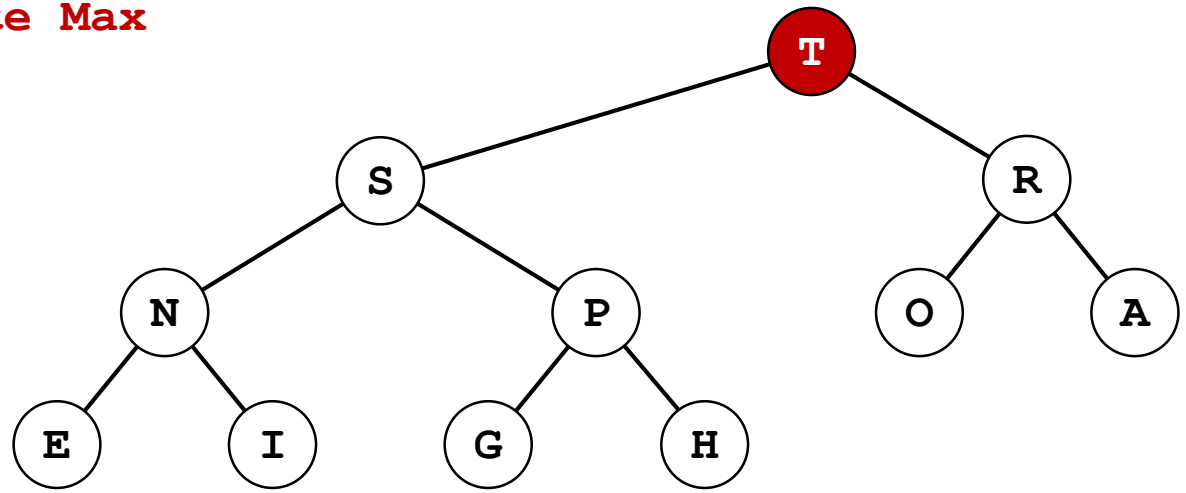


T	S	R	N	P	O	A	E	I	G	H

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max



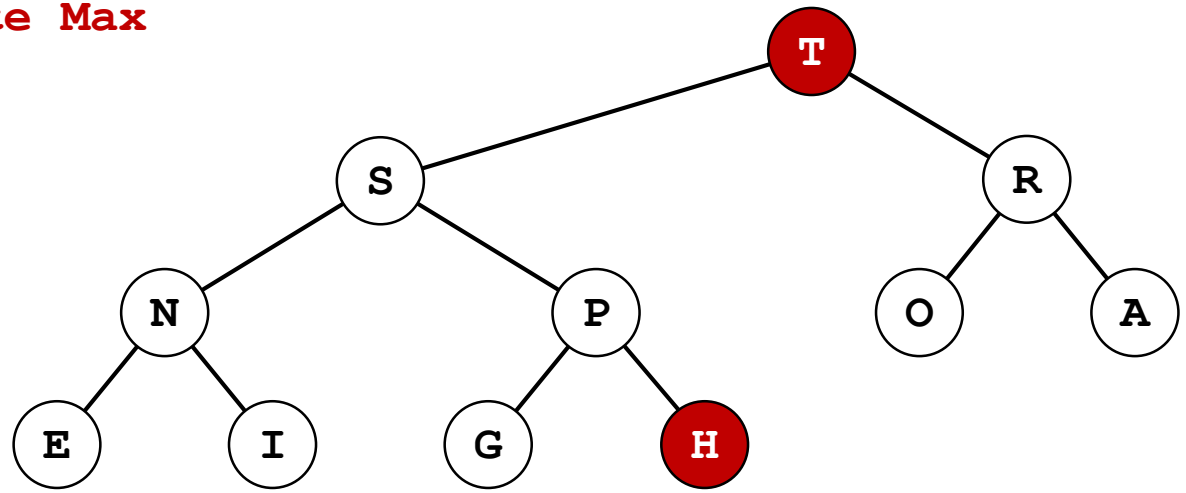
T	S	R	N	P	O	A	E	I	G	H

1

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
delete Max
```

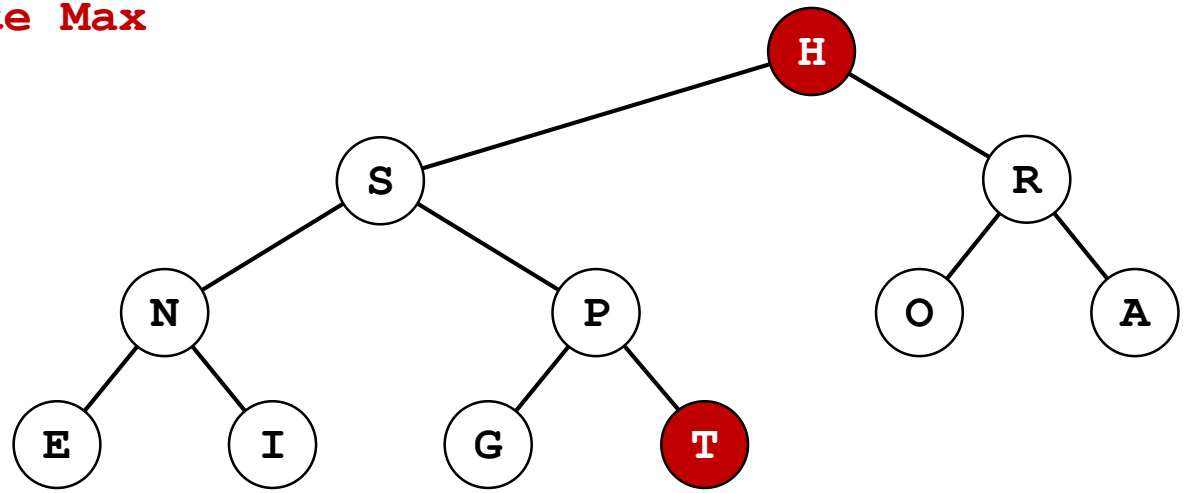


T	S	R	N	P	O	A	E	I	G	H
1										11

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

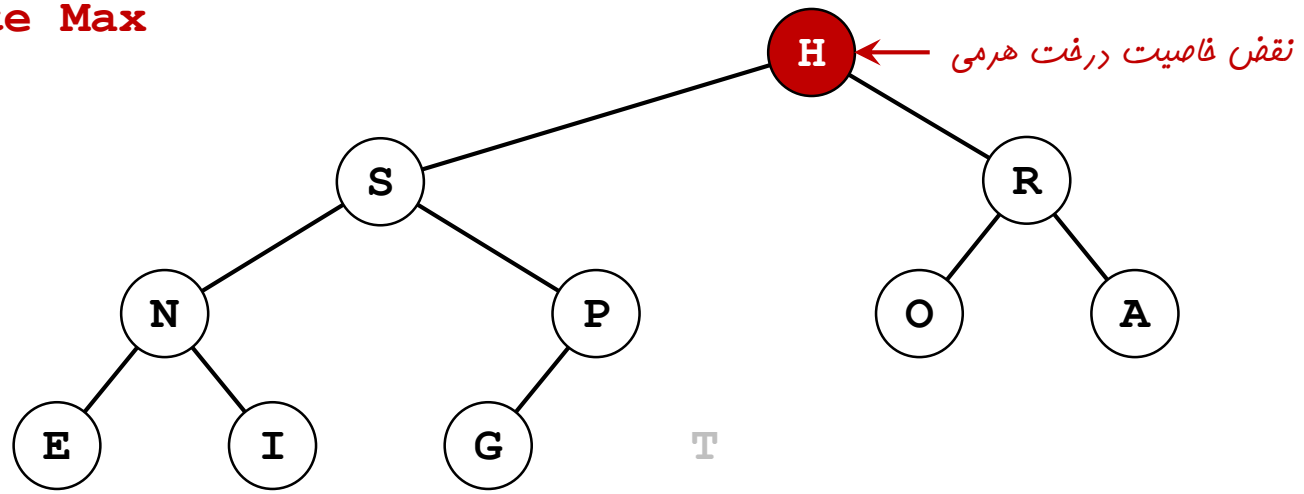


H	S	R	N	P	O	A	E	I	G	T
1										11

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

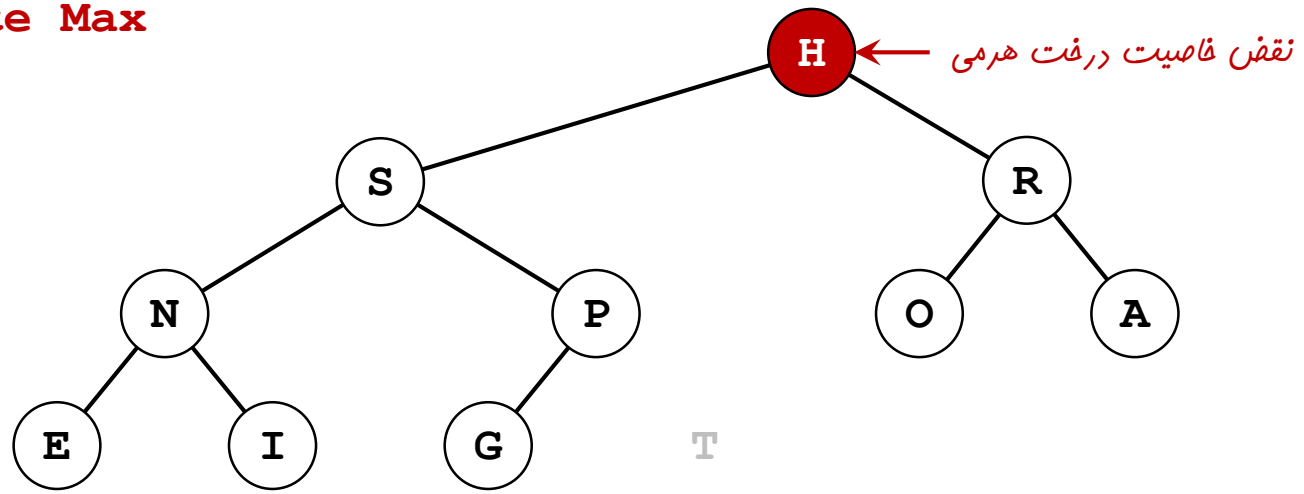


H	S	R	N	P	O	A	E	I	G	T
1										

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

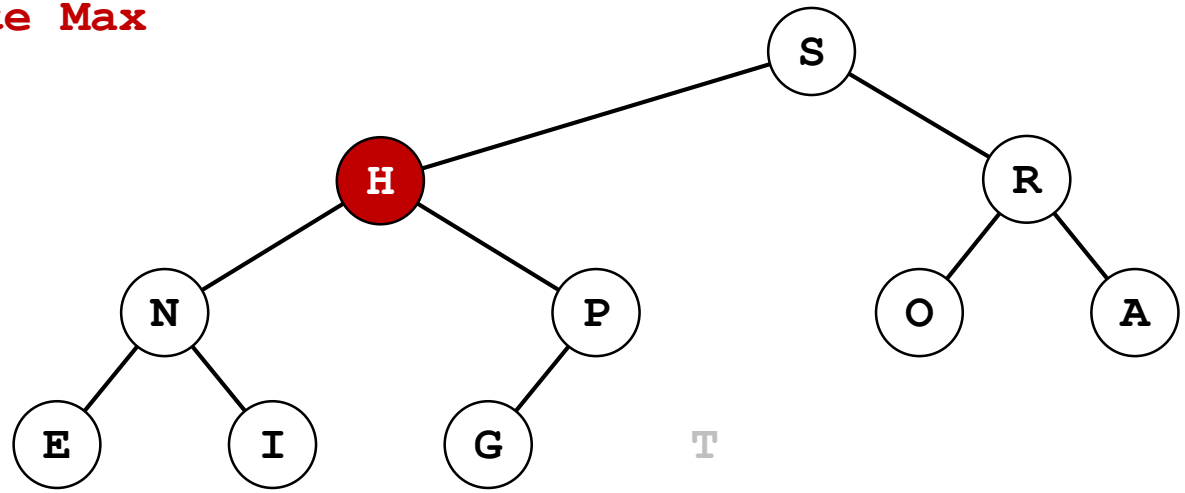


H	S	R	N	P	O	A	E	I	G	T
1	2									

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

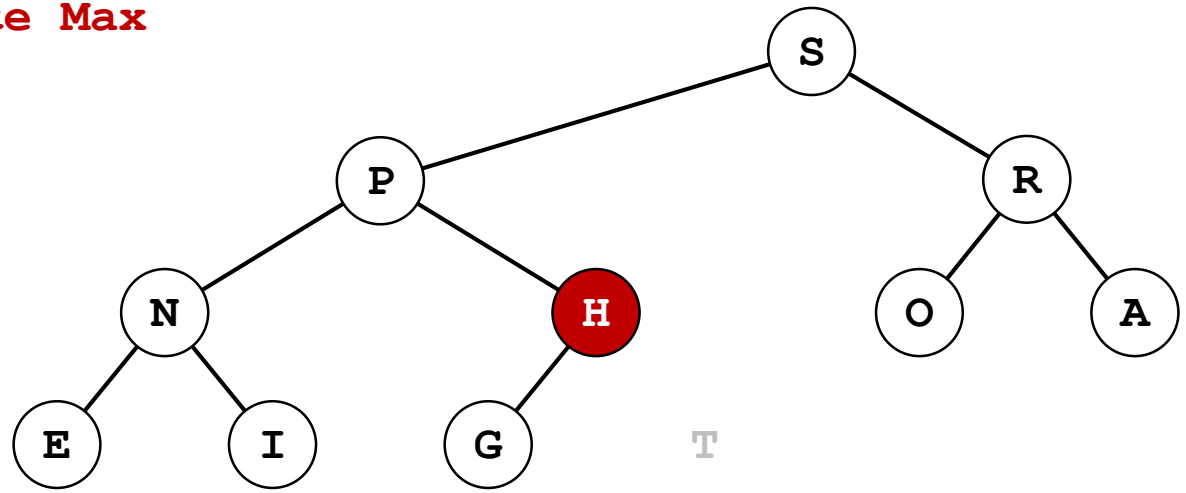


S	H	R	N	P	O	A	E	I	G	T
	2			5						

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
delete Max
```

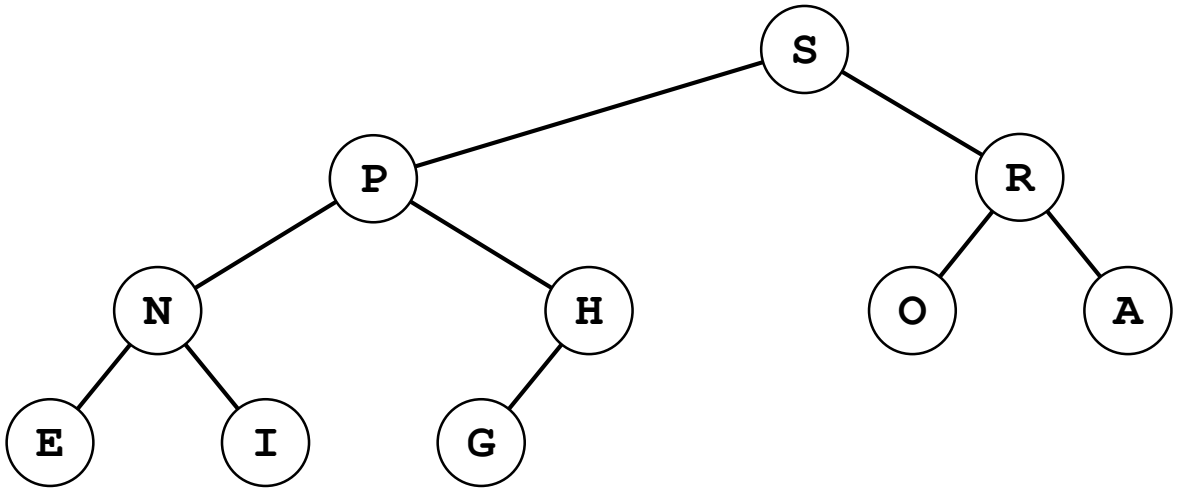


S	P	R	N	H	O	A	E	I	G	T

5

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

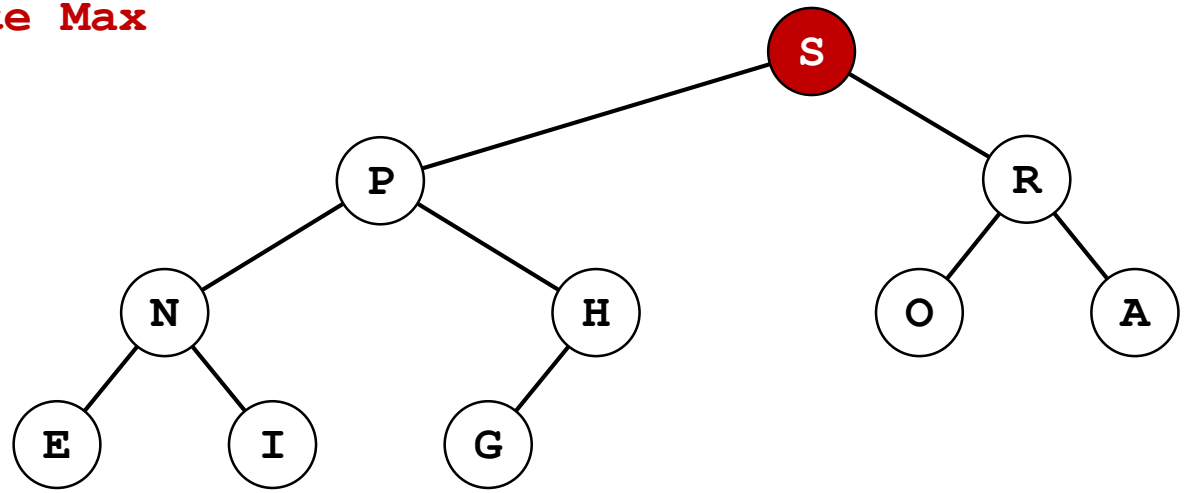


S	P	R	N	H	O	A	E	I	G	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max



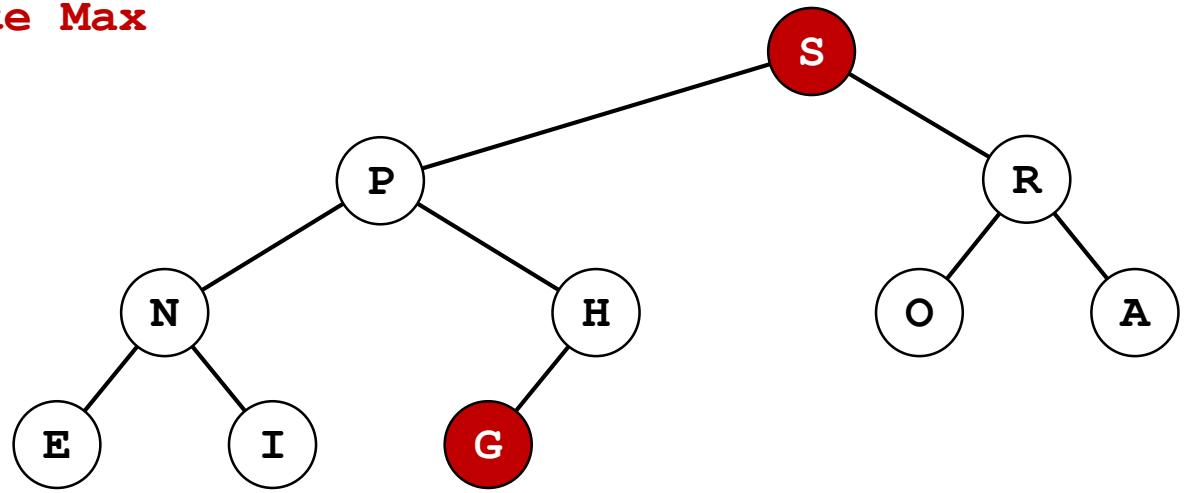
S	P	R	N	H	O	A	E	I	G	

1

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
delete Max
```

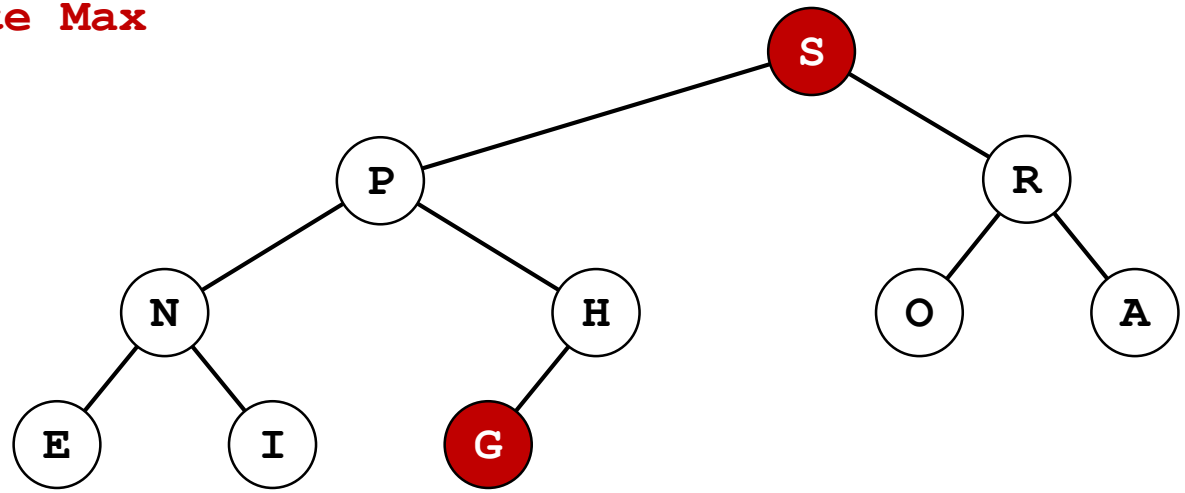


S	P	R	N	H	O	A	E	I	G	
1									10	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

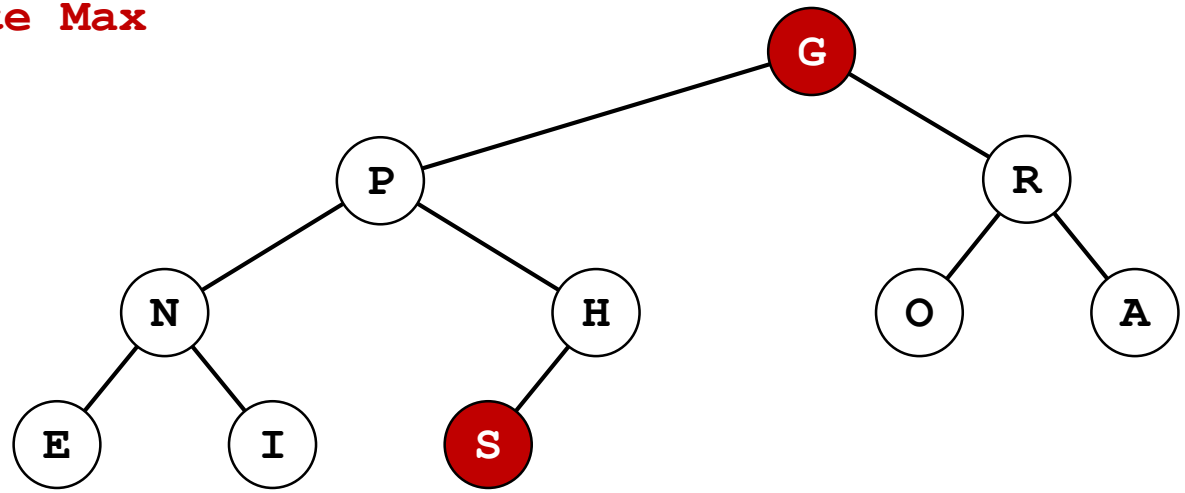


S	P	R	N	H	O	A	E	I	G	
1									10	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
delete Max
```

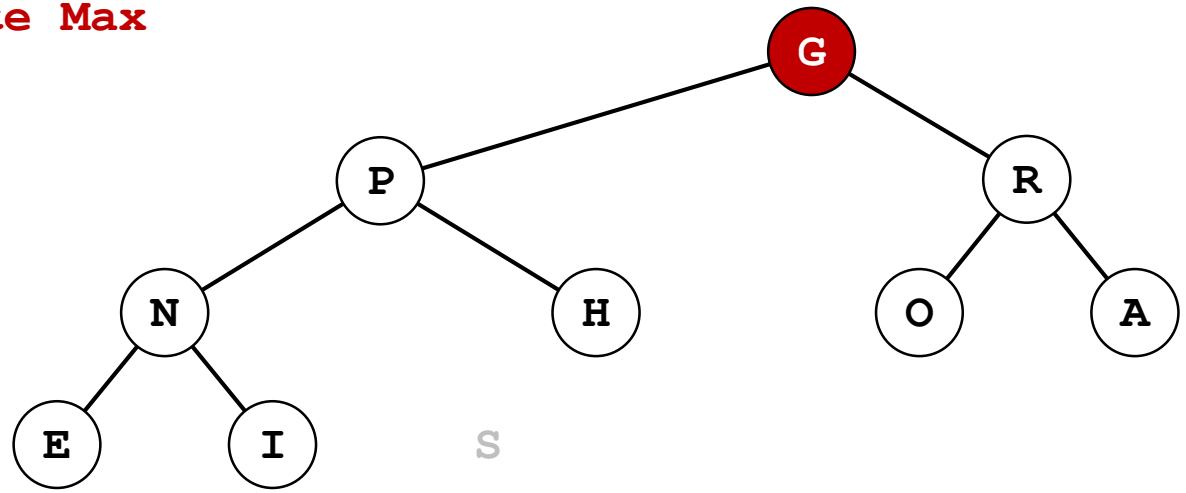


G	P	R	N	H	O	A	E	I	S	
1									10	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

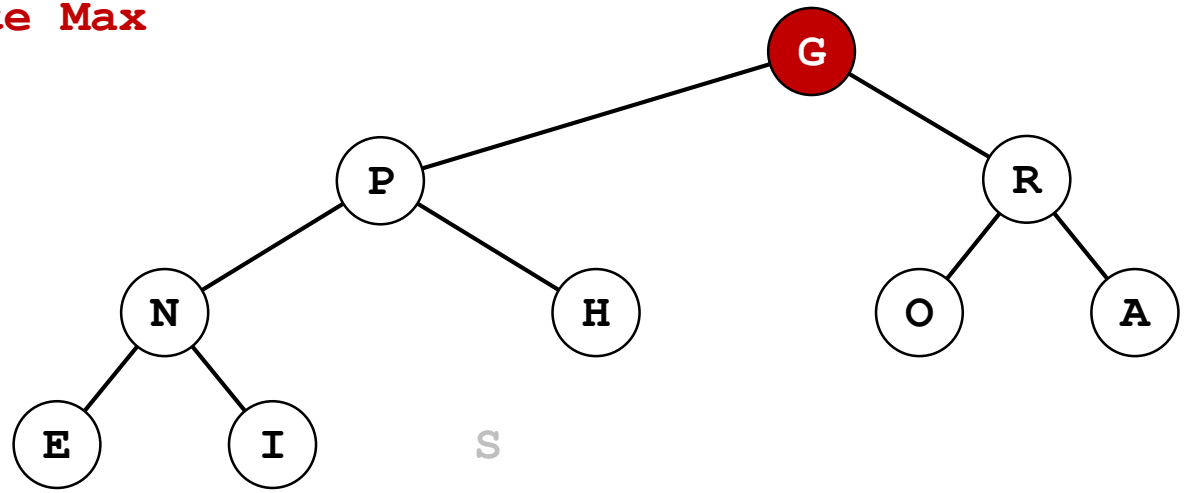


G	P	R	N	H	O	A	E	I	S	
1										

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

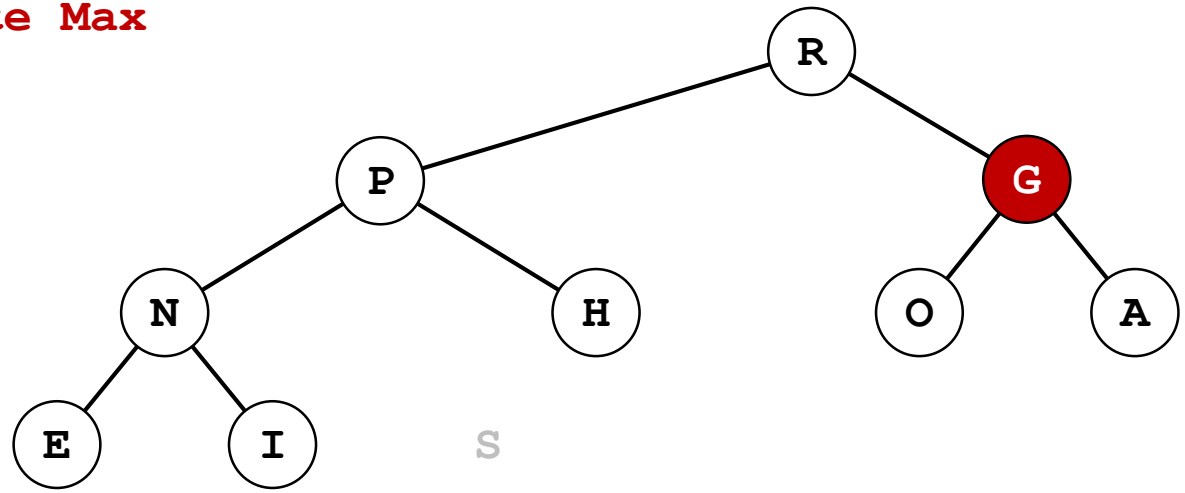


G	P	R	N	H	O	A	E	I	S	
1		3								

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

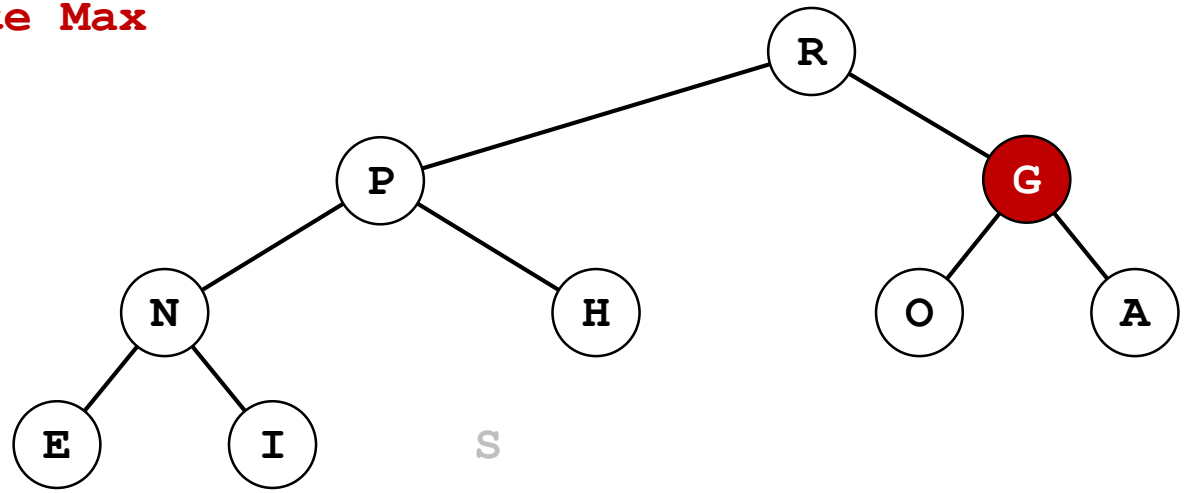


R	P	G	N	H	O	A	E	I	S	

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

delete Max

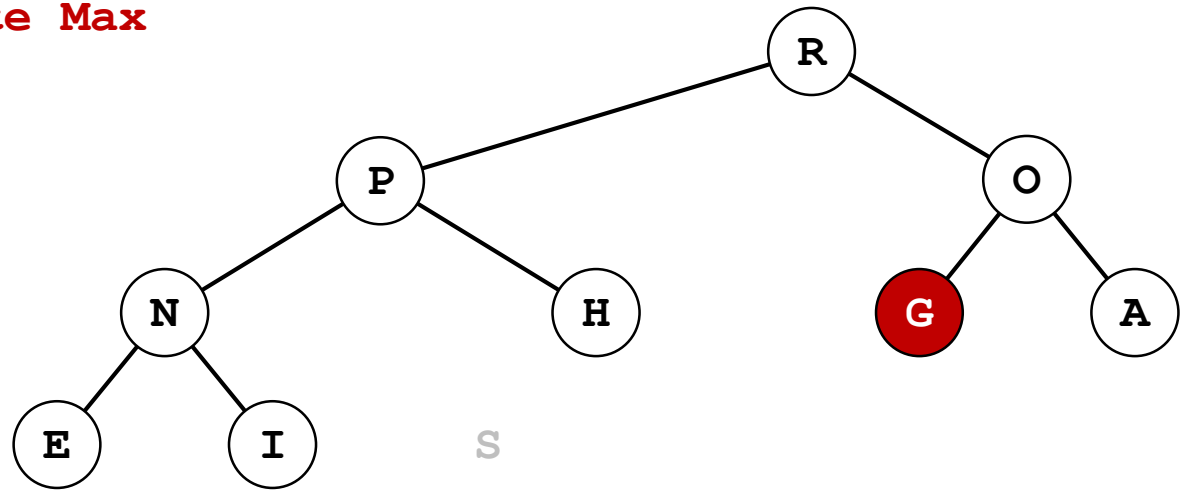


R	P	G	N	H	O	A	E	I	S	
		3			6					

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.

```
delete Max
```

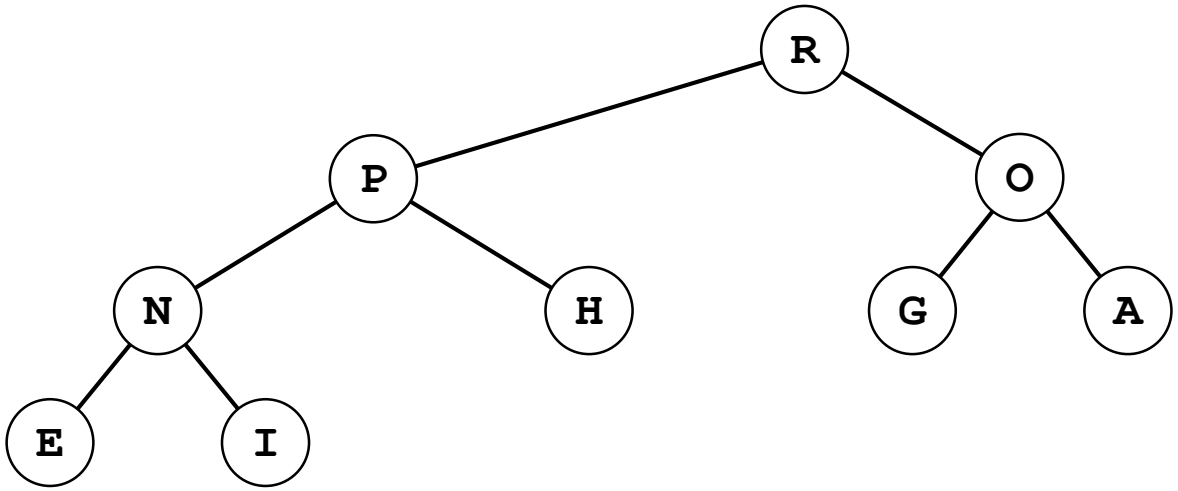


R	P	O	N	H	G	A	E	I	S	

6

اجرای نمایشی

- درج. اضافه کردن یک گره به انتهای درخت، حرکت به سمت بالا.
- حذف. جابجا کردن کلید ریشه با آخرین گره، حرکت به سمت پایین.



R	P	O	N	H	G	A	E	I		

درخت هرمی: پیاده‌سازی در جاوا


```
public class MaxPQ<Key extends Comparable<Key>> {
   private Key[] pq;      // pq[i] = ith element on pq
   private int N;         // number of elements on pq

   public MaxPQ(int capacity)
   {  pq = (Key[]) new Comparable[capacity + 1]; }

   public boolean isEmpty()
   {  return N == 0; }

   public void insert(Key x)
   {  /* see previous slides */  }

   public Key delMax()
   {  /* see previous slides */  }

   private void swim(int k)
   {  /* see previous slides */  }

   private void sink(int k)
   {  /* see previous slides */  }

   private boolean less(int i, int j)
   {  return pq[i].compareTo(pq[j]) < 0; }

   private void exch(int i, int j)
   {  Key t = pq[i]; pq[i] = pq[j]; pq[j] = t; }
}
```

مقایسه پیاده‌سازی مختلف صف اولویت

implementation	insert	delMax	max
Unordered array	1	N	N
Ordered array	N	1	1
binary heap	log N	log N	1
d-ary heap	$\log_d N$	$d \log_d N$	1
fibonacci	1	log N	1
impossible	1	1	1

چرا غیر ممکن؟

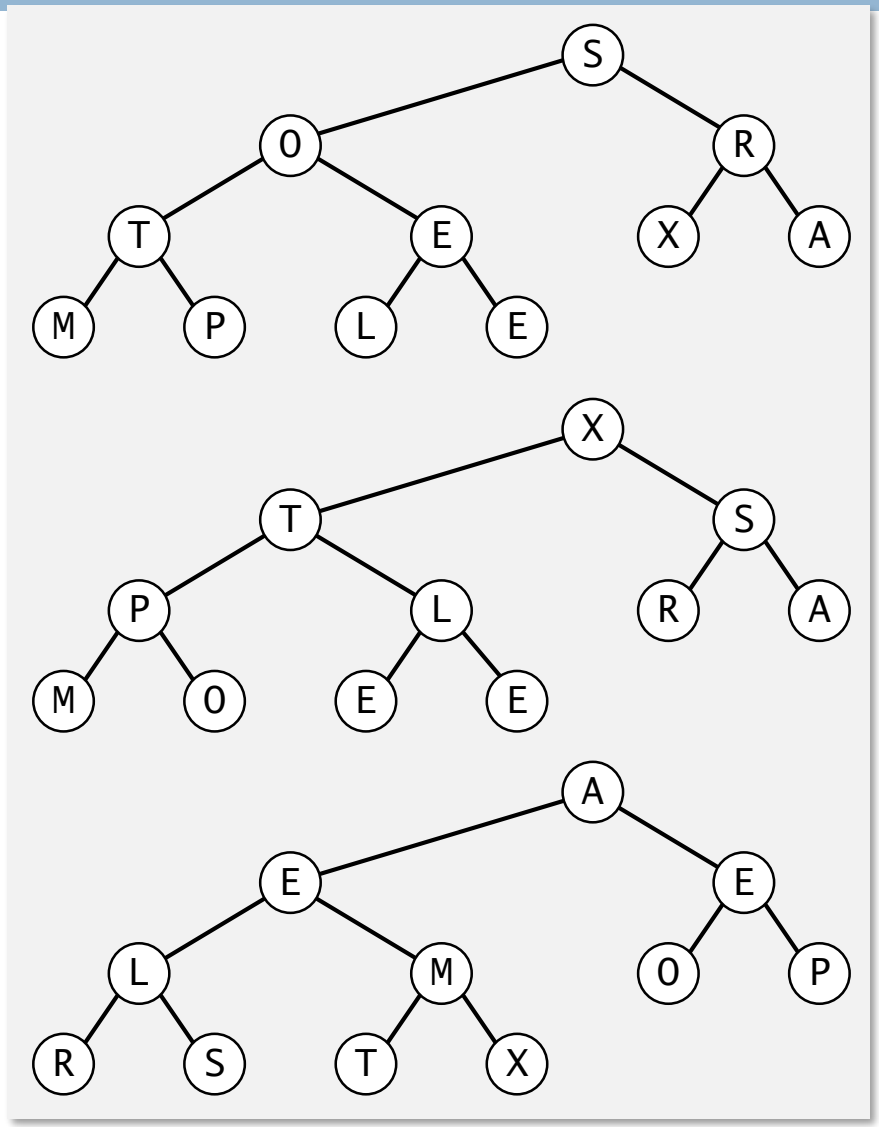
ملاحظات

- سرریز و پاریز.
  - پاریز: بروز یک استثنا در صورت حذف از یک صف اولویت تهی.
  - سرریز: دو برابر کردن اندازه آرایه.
- درخت هرمی کوچک.
  - جایگزینی تابع `less()` با تابع `greater()`
  - پیاده‌سازی تابع `greater()`
- عملیات دیگر.
  - حذف یک عنصر دلخواه.
  - تغییر اولویت یک عنصر موجود در صف.

مرتب‌سازی هرمی

مرتب‌سازی هرمی

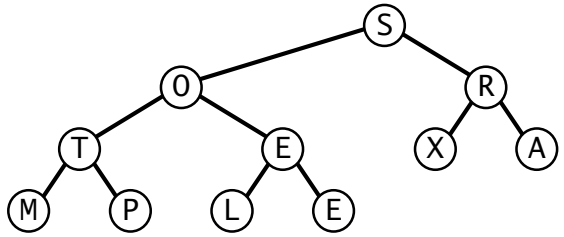
- طرح اصلی برای مرتب‌سازی درجا.
  - ایجاد یک درخت هرمی بزرگ با **N** کلید
  - حذف مکرر بزرگ‌ترین کلید

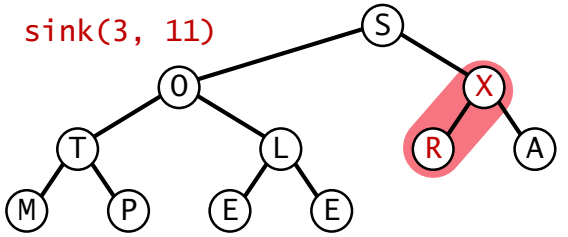


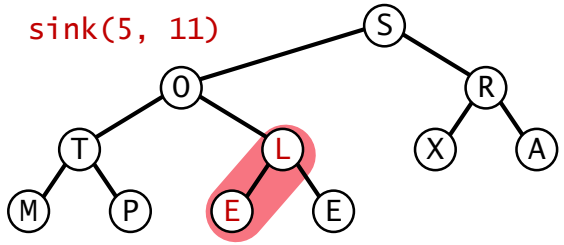
مرتب‌سازی هرمی: ساختن هرم

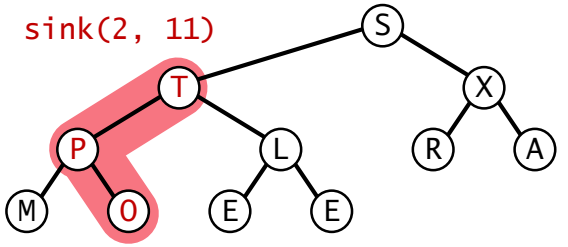
- گذر اول. ساختن هرم به صورت پایین به بالا.

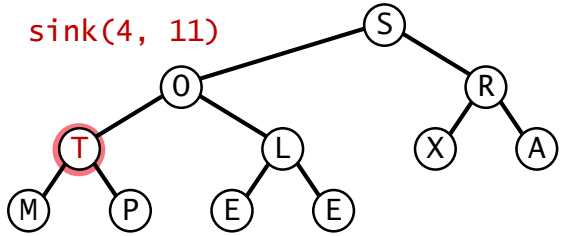
```
for (int k = N / 2; k >= 1; k--)
   sink(a, k, N);
```

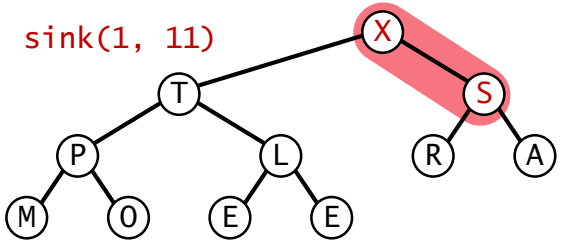








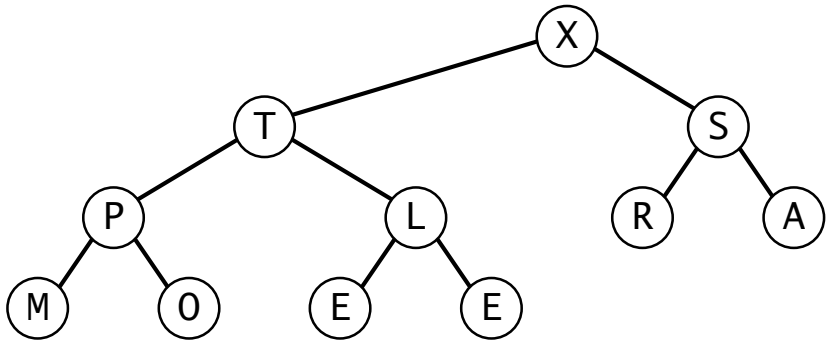




مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

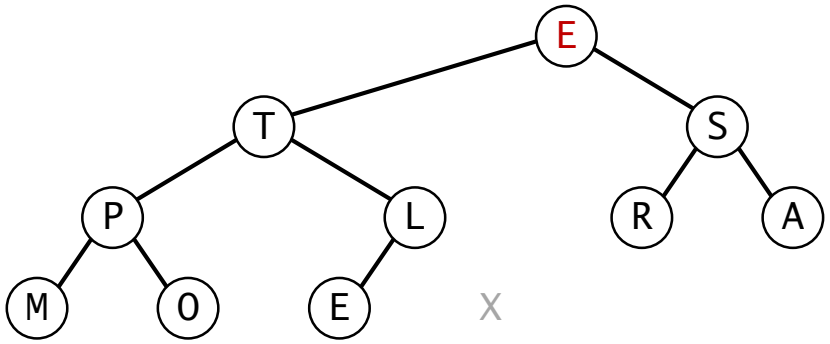
- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

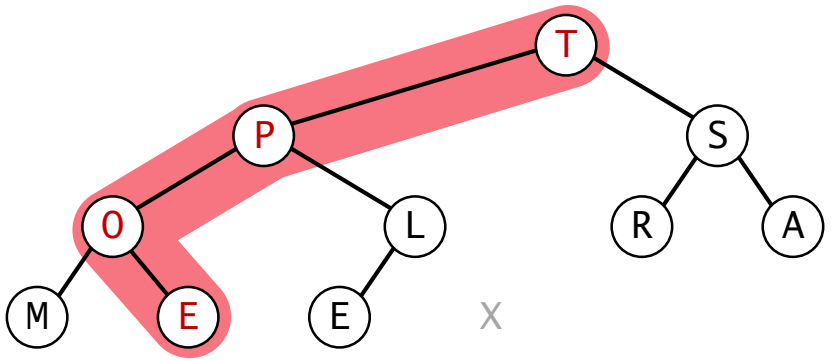


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

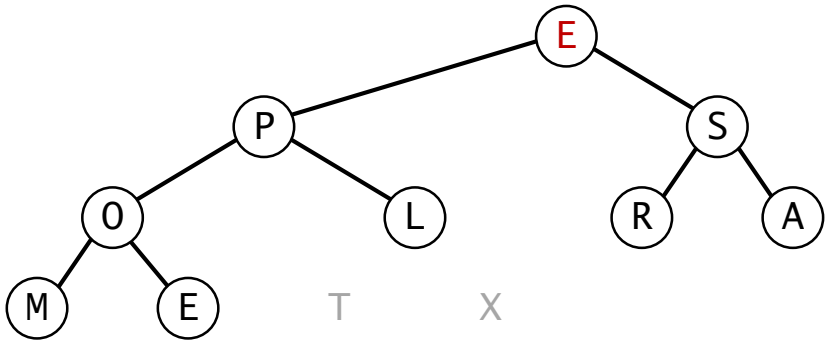


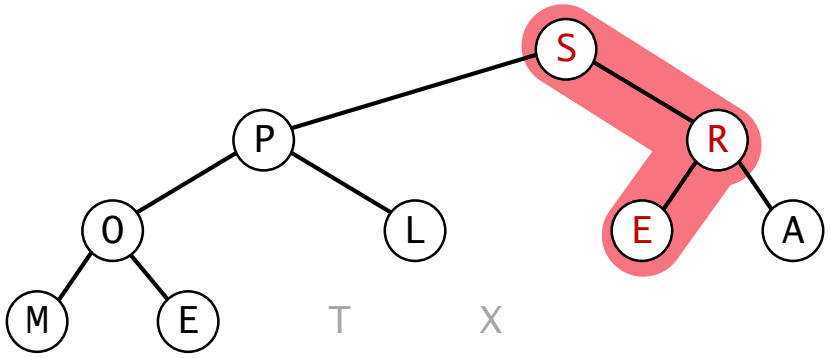


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

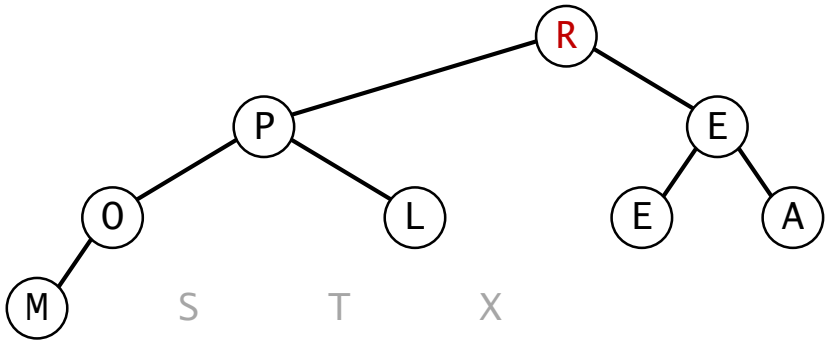


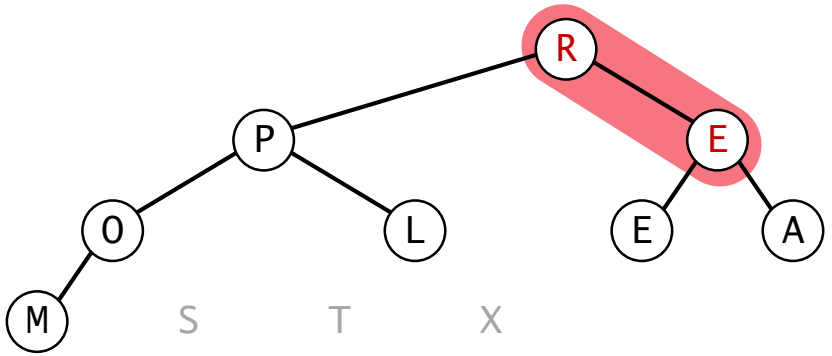


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

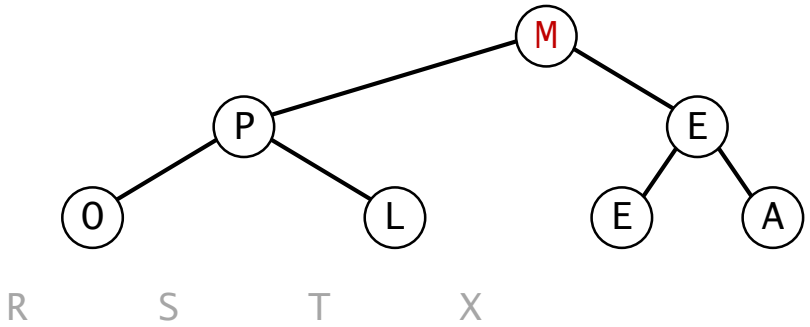


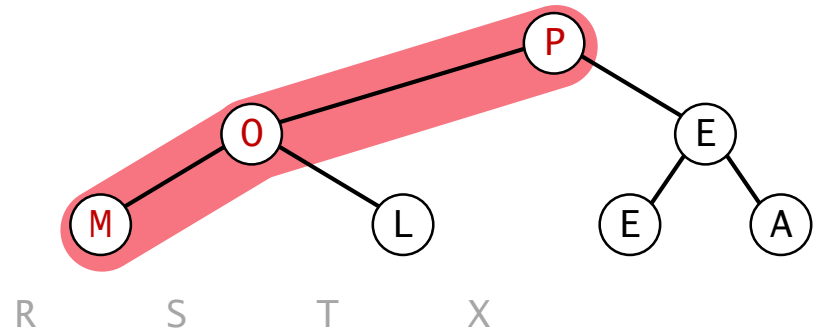


مرتب‌سازی هرمی: مرتب کردن

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

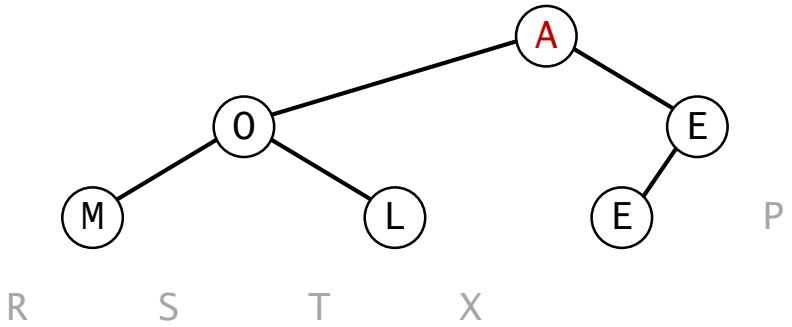


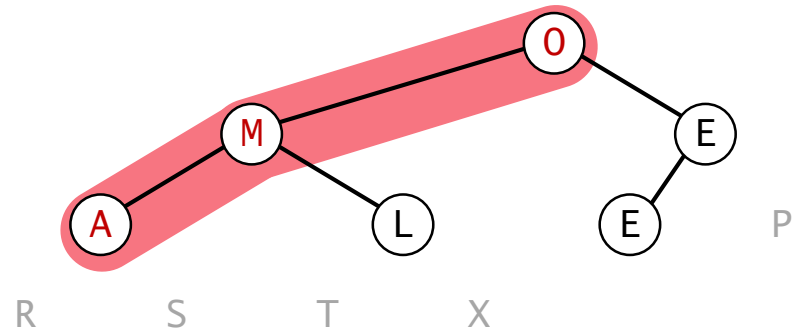


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

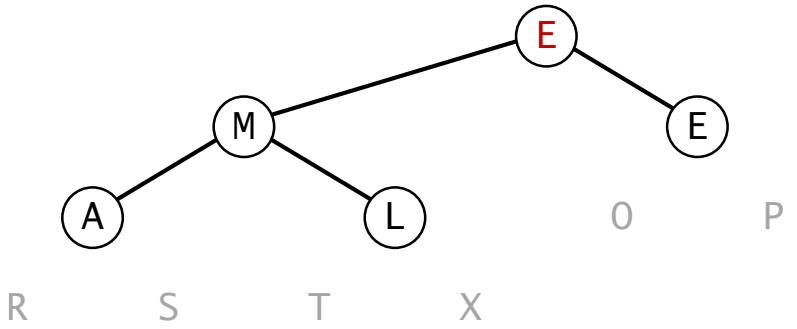


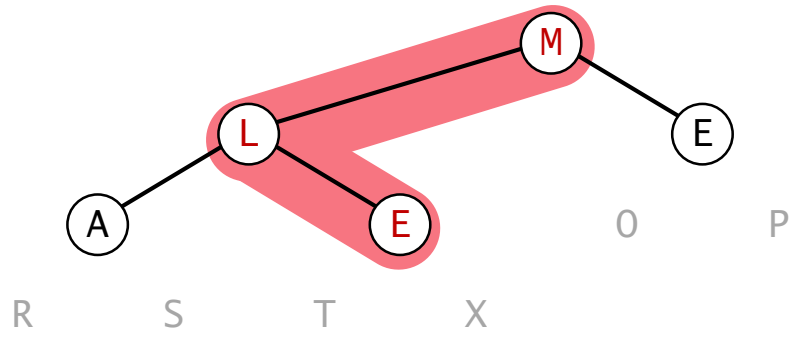


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

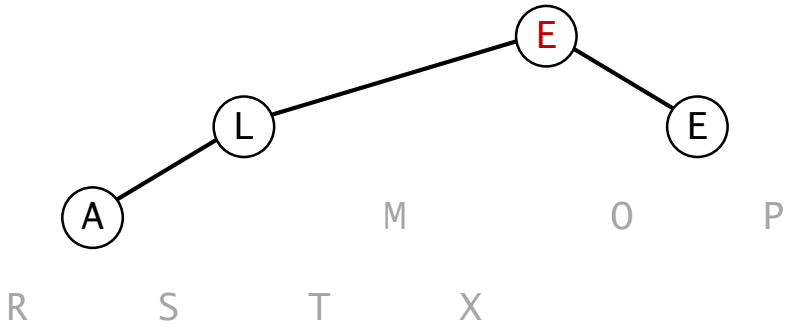


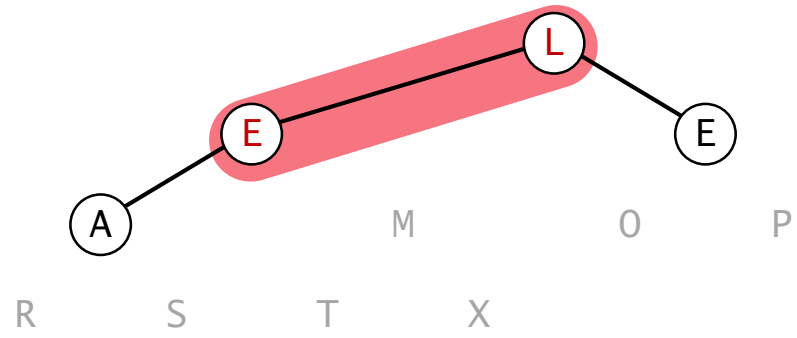


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

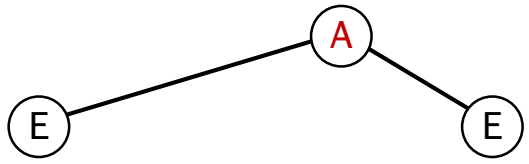




مرتب‌سازی هرمی: مرتب کردن

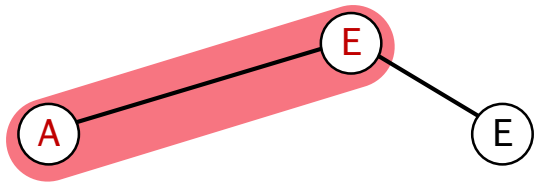

```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.



L M O P

R S T X



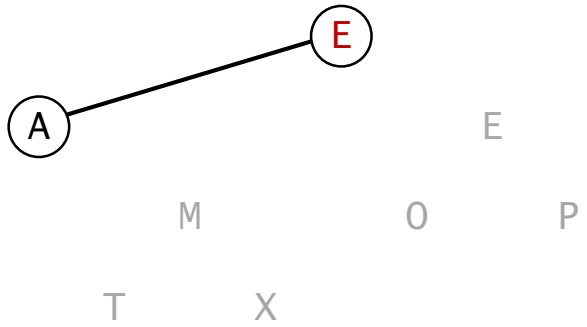
L M O P

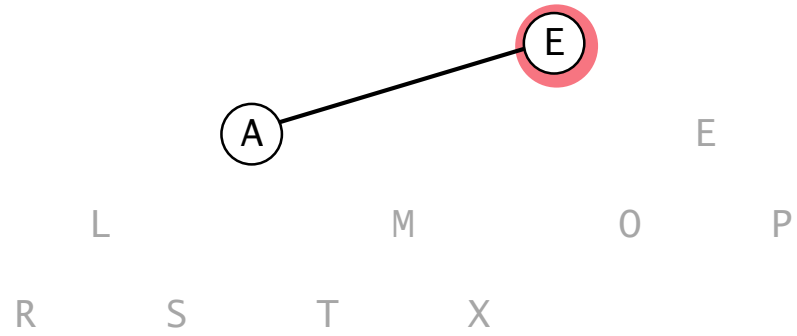
R S T X

مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

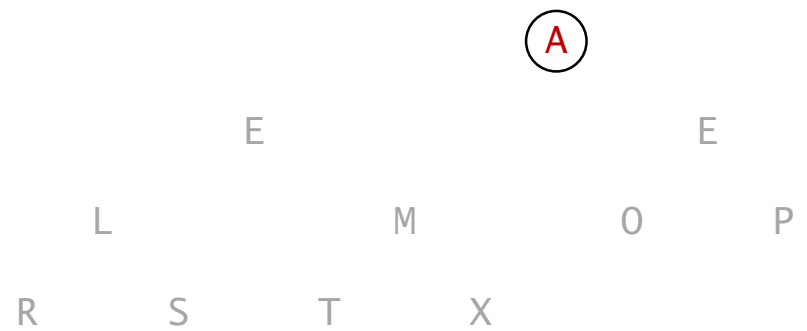


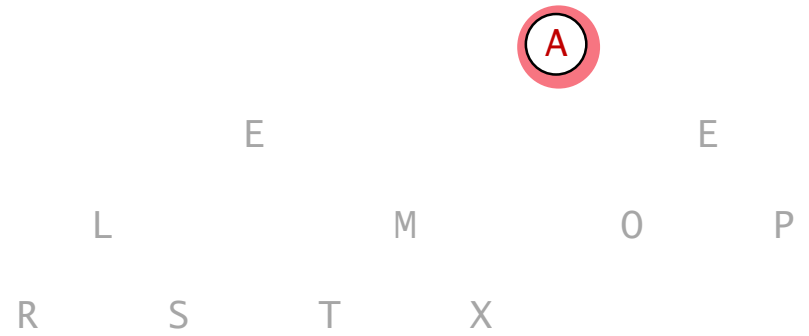


مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.

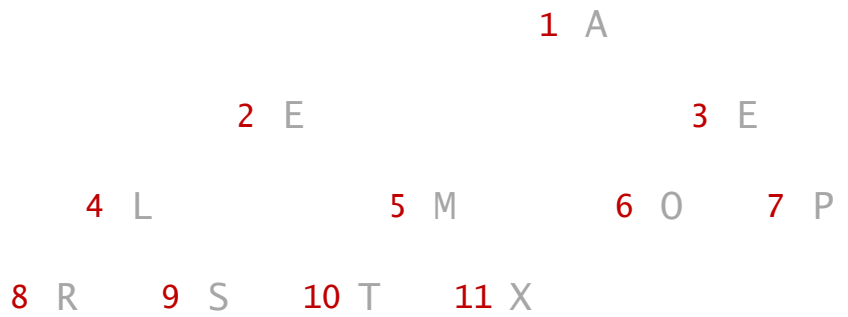




مرتب‌سازی هرمی: مرتب کردن


```
while (N > 1)
{
   exch(a, 1, N--);
   sink(a, 1, N);
}
```

- گذر دوم.
  - حذف بزرگ‌ترین عنصر به طور مکرر.
  - نگهداشتن عنصر حذف شده در آرایه.



i	0	1	2	3	4	5	6	7	8	9	10	11
a[i]	-	A	E	E	L	M	O	P	R	S	T	X

مرتب‌سازی هرمی: پیاده‌سازی در جاوا


```
public class HeapSort
{
   public static void sort(Comparable[] pq)
   {
      int N = pq.length;
      for (int k = N / 2; k >= 1; k++)
         sink(pq, k, N);
      while (N > 1)
      {
         exch(pq, 1, N--);
         sink(pq, 1, N);
      }
   }

   private static void sink(Comparable[] pq, int k, int N)
   {  /* as before */  }

   private static boolean less(Comparable[] pq, int i, int j)
   {  /* as before */  }

   private static void exch(Comparable[] pq, int i, int j)
   {  /* as before */  }
}
```

تحلیل مرتب‌سازی هرمی

- گزاره. ساختن هرم به حداکثر **2N** مقایسه و جابجایی نیاز دارد.
- گزاره. مرتب‌سازی هرمی به حداکثر **2N lg N** مقایسه و جابجایی نیاز دارد.

- اهمیت. یک روش مرتب‌سازی درجا که در بدترین حالت از مرتبه‌ی **N lg N** است.
  - مرتب‌سازی ادغامی: خیر، مصرف حافظه‌ی کمکی خطی
  - مرتب‌سازی سریع: خیر، در بدترین حالت از مرتبه درجه دوم
  - مرتب‌سازی هرمی: بله!

- سخن آخر. مرتب‌سازی هرمی از لحاظ حافظه و زمان بهینه است، اما:
  - در حالت متوسط اندکی کندتر از مرتب‌سازی سریع است.
  - به خوبی از حافظه‌ی کش استفاده نمی‌کند.
  - پایدار نیست!