Cheatography

Data Structures Cheat Sheet

by TowChow23 via cheatography.com/28831/cs/8484/

Push

```
public void push(T newEntry) {
                // Add to
beginning of chain:
                try {
                        Lin -
ked Sta ck.Node newNode = new
Linked Sta ck.N od e(n ewE -
ntry);
                        new -
Nod e.next = firstNode; // Make
new node reference rest of chain
                       //
(firstNode is null if chain is
empty)
                        fir -
stNode = newNode; // New node is
at beginning of chain
                        num -
ber OfE ntr ies++;
                } catch (OutOf -
Mem ory Error e) {
                        throw
new Illega lSt ate Exc ept -
ion();
                }
                 //r eturn true;
        }
```

Pop

```
@Override
        public T pop() {
                T result = null;
                if (firstNode !=
null) {
                        result =
firstN ode.data;
                        fir -
stNode = firstN ode.next; //
Remove first node from chain
                        num -
ber OfE ntr ies--;
                } else {
                        throw
new NoSuch Ele men tEx cep -
tion();
                }
                 return result;
        }
```

Induction Proofs

Claim: *for any* $n \geq 1$, $1+2+3+4+\cdots+n=\frac{n\cdot(n+1)}{2}$

Proof:

- Base case: $n=1$ $\quad 1=\frac{1\cdot 2}{2}$ ✔
- Induction step:

  for any $k \geq 1$, *if* $\quad 1+2+3+4+\cdots+k=\frac{k\cdot(k+1)}{2}$

  then $\quad 1+2+3+4+\cdots+k+(k+1)=\frac{(k+1)\cdot(k+2)}{2}$

Recursive Fern

```
public void drawFern(double x,
double y, double angle, double
size) {
if (size > 1.0) { // STOP if
size <= 1.0!
double[] end;
double length = size * 0.5;
end = drawSt em(x, y, angle,
length); // private method
double smaller = size * 0.5; //
SMALLER!
drawFe rn( end[0], end[1],
angle+60, smaller);
drawFe rn( end[0], end[1],
angle, smaller);
drawFe rn( end[0], end[1],
angle-60, smaller);
}
}
```

Recursive Binary

Selection Sort (cont)

```
>               a[minPos] = a[i];
                a[i] = temp;
        }
    }
```

Shell Sort

```
public static <T extends
Comparable<? super T>>
        void shellS ort(T[] a) {
                int gap =
a.length / 2;
                 while (gap >= 1)
{
                        if (gap %
2 == 0) ++gap;
                        for (int
i = gap; i < a.length; ++i) {
                                i
nt p = i;
                                T
temp = a[p];
                  while (p >=
gap && a[p-ga p].c om par eTo -
(temp) > 0) {
                                -
     a[p] = a[p-gap];
                                -
       p -= gap;
                                }
                                a
[p] = temp;
                        }
                        gap /= 2;
                }
        }
```

Insertion

Top

```
public T top() {
               if (firstNode !=
null) {
                        return
firstN ode.data;
               }
                throw new
NoSuch Ele men tEx cep tio -
n("Stack is Empty");
       }
```

```
public static <T> int
binaryFind(Comparable<T> item,
T[] v, int lo, int hi) {
               if (lo > hi) {
return -1; }
               int mid = lo +
(hi - lo) / 2;
               if (item.c om -
par eTo (v[ mid]) < 0) {
                         return
binary Fin d(item, v, lo, mid -
1);
               } else if
(item.c om par eTo (v[ mid]) >
0) {
                         return
binary Fin d(item, v, mid+1,
hi);
               } else { return
mid; } // found it!
       }
```

Selection Sort

```
public static <T extends
Comparable<? super T>>
                        void
select ion Sor t(T[] a) {
               for (int i = 0; i
< a.length - 1; ++i) {
                        int
minPos = i;
                        for (int
j = i + 1; j < a.length; j++) {
                                i
f (a[j].c om par eTo (a[ min -
Pos]) < 0) {
                                -
        minPos = j;
                                }
                        }
                                T
temp = a[minPos];
```

```
public static <T extends
Comparable<? super T>>
       void insert ion Sor t(T[]
a) {
               for (int i = 0; i
< a.length - 1; ++i) {
                        int p = i
+ 1;
                        T temp =
a[p];
                         while (p
> 0 && a[p-1].co mpa reT o(temp)
> 0) {
                                a
[p] = a[p-1];
                                -
-p;
                        }
                        if (p >
0) // count the last a[p-1]
comparison
                        a[p] =
temp;
               }
       }
```

By **TowChow23**

Published 23rd June, 2016.
Last updated 23rd June, 2016.