

Chapter 5 notes

What is algorithm:

It's a clearly specified instructions the computer will follow to solve a problem
Want to know which way is faster.

With a method to do more if you pass it a bigger number than it takes longer

1000 runs at 684658

2000 runs at 264249

We expected it to run longer because we gave it 2x as much but the second time ran faster

This is a guess

Since it is in the jvm has already loaded the information it is faster the second time

When trying to figure the times we are using the mean but some outliers throw us off so
then we use the medium to see if it could improve

If you can run in $N \log N$ that is the goal that we want to go for

If you have some bad input then some will be better. Like quadratic or cubic

Linear is usually always the fastest

Moderate inputs: linear is faster, $\log n$ next, then quadratic, then cubic

Constant will be better than linear

How much better will be better

Cubic was the worst so $10 N^3 + N^2 + N$

Log represents a function whose dominant term is N . ($N \log$)

Big-Oh notation is the term used when we only discuss the dominant term

C	Constant
Log N	Logarithmic
$\text{Log}^2 N$	Log-Squared
N	Linear
$N \log N$	$N \log N$
N^2	Quadratic
N^3	Cubic

2^N	Exponential
-------	-------------

Theorem 5.1 red ball is lower then number and blue is more the highest
 $1 \leq i \leq k \leq j$

Example: pull out red, 4, 8
Get 4, 4, 8

5.4 HW
 X^2 , x maybe more

- 5.23 - Let's define 'larger' to mean 'has more employees.' Further note: When a company swallows up Joe's company, they don't fire anyone, they keep all the employees, including Joe. For example, if Joe works in a company of 10 employees and that company gets swallowed up by a second company with 11 employees, the resulting company will have $10+11 = 21$ employees.

Unlucky Joe Joe works for a company with n employs. Joe company always gets bought out by a company with more employs. So how many companies has he been bought by.

1

1

(2)

3

3

4

5

6

100

1

3

7

15

31

63

1

3

7

15

31

63

127

255

511

1023

2047

4095

8191

16383

32767

65535

131071

262143

524287

1048575

2097151

4194303

8388607

16777215

33554431

67108863

134217727

268435455

536870911

1073741823

2147483647

4294967295

8589934591

17179869183

34359738367

68719476735

137438953471

274877906943

549755813887

1099511627775

2199023255551

4398046511103

8796093022207

17592186044415

35184372088831

70368744177663

140737488355327

281474976710655

562949953421311

1125899906842623

2251799813685247

4503599627370495

9007199254740991

18014398509481983

36028797018963967

72057594037927935

144115188075855871

288230376151711743

576460752303423487

1152921504606846975

2305843009213693951

4611686018427387903

9223372036854775807

18446744073709551615

36893488147419103231

73786976294838206463

147573952589676412927

295147905179352825855

590295810358705651711

1180591620717411303423

2361183241434822606847

4722366482869645213695

9444732965739290427391

18889465931478580854783

37778931862957161709567

75557863725914323419135

151115727451828646838271

302231454903657293676543

604462909807314587353087

1208925819614629174706175

2417851639229258349412351

4835703278458516698824703

9671406556917033397649407

19342813113834066795298815

38685626227668133590597631

77371252455336267181195263

154742504910672534362390527

309485009821345068724781055

618970019642690137449562111

1237940039285380274899124223

2475880078570760549798248447

4951760157141521099596496895

9903520314283042199192993791

19807040628566084398385987583

39614081257132168796771975167

79228162514264337593543950335

158456325028528675187087900671

316912650057057350374175801343

633825300114114700748351602687

1267650600228229401496703205375

2535301200456458802993406410751

5070602400912917605986812821503

10141204801825835211973625643007

20282409603651670423947251286015

40564819207303340847894502572031

81129638414606681695789005144063

162259276829213363391578010288127

324518553658426726783156020576255

649037107316853453566312041152511

1298074214633706907132624082305023

2596148429267413814265248164610047

5192296858534827628530496329220095

10384593717069655257060992658440191

20769187434139310514121985316880383

41538374868278621028243970633760767

83076749736557242056487941267521535

166153499473114484112975882535043071

332306998946228968225951765070086143

664613997892457936451903530140172287

1329227995784915872903807060280344575

2658455991569831745807614120560689151

5316911983139663491615228241121378303

10633823966279326983230456482242756607

21267647932558653966460912964485513215

42535295865117307932921825928971026431

85070591730234615865843651857942052863

170141183460469231731687303715884105727

340282366920938463463374607431768211455

680564733841876926926749214863536422911

136112946768375385385349842972707284583

272225893536750770770699685945414569167

544451787073501541541399371890829138335

1088903574147003083082798743781658276671

2177807148294006166165597487563316553343

4355614296588012332331194975126633106687

8711228593176024664662389950253266213375

17422457186352049329324779900506532426751

34844914372704098658649559801013064853503

69689828745408197317299119602026129707007

139379657490816394634598239204052259414015

278759314981632789269196478408104518828031

557518629963265578538392956816209037656063

1115037259926531157076785913632418075312127

2230074519853062314153571827264836150624255

4460149039706124628307143654529672301248511

892029807941224925661428730905

n.

To do linear only get 1 for loop. We need to be more clever we will need more memory, variables, storage space. Time/space tradeoff.

Purpose is to be able to look at code and say what kind of growth is it like linear for ex.

General Big-oh rules:

Big-oh upper bound might be the same \leq

Big-Omega Lower bound \geq always bigger linear

Big-theta: both bounds are the same \Rightarrow always linear

Little-oh upper bound - will be the smaller $<$

5.5 the logarithm

Def for any $B, N > 0, \log_b N = k$ if $B^k = N$

$\log_b N$

When things halves or double that how we figure things are log.

Log of cuttings 1000

500

250

125

65

32

15

7

3

1

10 times

5.6 static search

Int $x = 50$ array of numbers return position in x or say its not their

Linear looking from left to right

5, 15, -8, 2, 4, 9, 5

Use sorting

Use sequential search

What is the cost of an unsuccessful search? Looked at all them didn't find it

What is the worst case of a successful search? Find it at last

$N-1$

Average case of successful search? Would $1/N$

Linked list:

```
Class ListNode{
    Object element;
    ListNode next;
}
```

Class name also name as variable? It is like recursion, java uses pointers. So this line

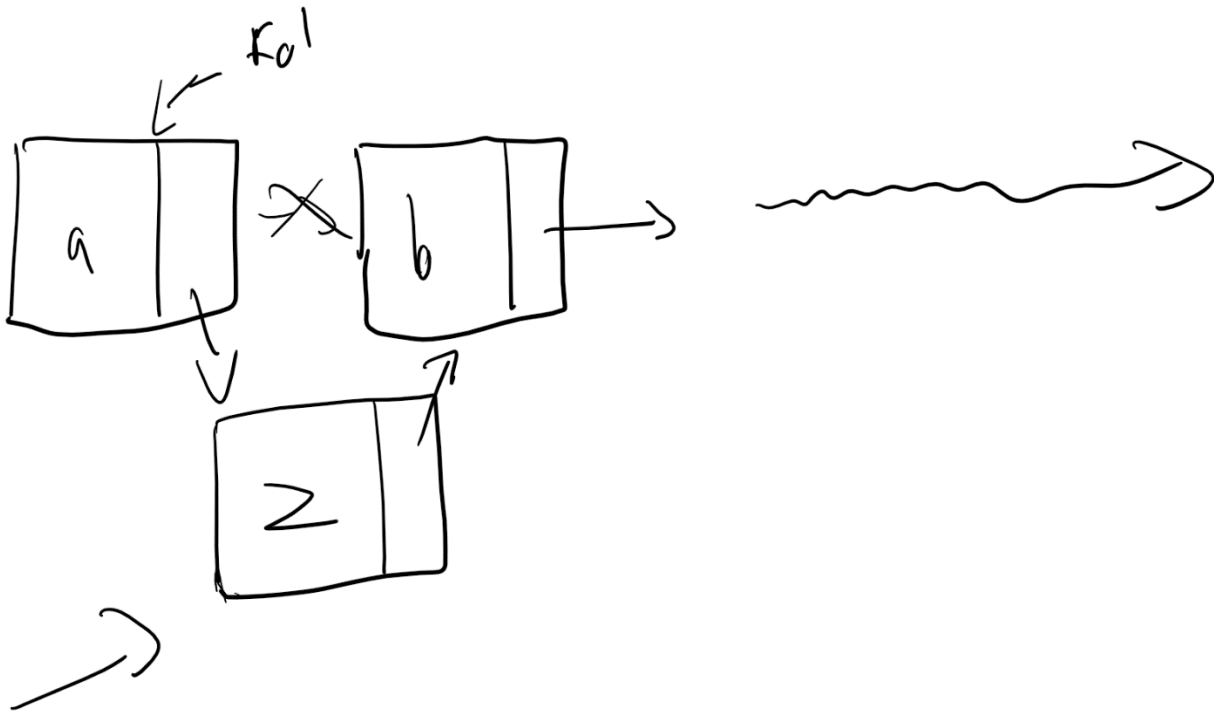
ListNode next;

Is just a pointer to a ListNode

To get to the end of linked list bigOh is N

For an array to get to the end bigOh is constant you go straight through it.

add link list between a & b



temp = New ListNode ();

temp.next = fol.next

fol.next = temp

Constant BigOh if you are putting stuff at the front for link list

Array for adding to the front would be BigOh N

Where would you want to use array or link list

Keep track of people by there height.

New person average middle height would matter which one you picked

If you have a group and you know the tallest person comes in first and everyone else is after you want to add shortest in first slot. Best would be link list because you know you can add to the front rather than array.