

HW 4

Derrick DeBose

ID# 1536268

8.1-1) If the comparison sort is

already in sorted order then $n-1$ comparisons
will be made and that means the smallest
possible depth of a leaf in a decision tree is $\boxed{n-1}$.

$$8.1-2) \sum_{k=1}^n \lg k$$

$$\lg(n!) = \Theta(n \lg n)$$

Proof: Big O

$$\sum_{k=1}^n \lg k \leq \sum_{k=1}^n \lg n = n \lg n = O(n \lg n)$$

Proof: Big Omega

$$\sum_{k=1}^n \lg k = \sum_{k=1}^{\lfloor n/2 \rfloor} \lg k + \sum_{k=\lfloor n/2 \rfloor + 1}^n \lg k$$

$$\geq \sum_{k=\lfloor n/2 \rfloor}^n \lg k$$

$$\geq \sum_{k=\lfloor n/2 \rfloor}^n \lg n/2$$

$$\geq (n/2) \lg(n/2)$$

$$= \Omega(n \lg n)$$

$$\boxed{\therefore \sum_{k=1}^n \lg k = \Theta(n \lg n)}$$

8.2-1)

A =

6	0	2	0	1	3	4	6	1	3	2
---	---	---	---	---	---	---	---	---	---	---

C =

0	1	2	3	4	5	6
2	2	2	2	1	0	2

c =

0	1	2	3	4	5	6
2	4	6	8	9	9	11

B =

1	2	3	4	5	6	7	8	9	10	11
					2					

c =

2	4	5	8	9	9	11
---	---	---	---	---	---	----

B =

		1	2	3						
--	--	---	---	---	--	--	--	--	--	--

c =

2	3	5	7	9	9	11
---	---	---	---	---	---	----

B =

		1	2	3	3	4				
--	--	---	---	---	---	---	--	--	--	--

c =

2	3	5	6	8	9	10
---	---	---	---	---	---	----

B =

	0	1	1	2	2	3	3	4		6
--	---	---	---	---	---	---	---	---	--	---

c =

1	2	4	6	8	9	10
---	---	---	---	---	---	----

SORTED! B =

0	0	1	1	2	2	3	3	4	6	6
---	---	---	---	---	---	---	---	---	---	---

c =

6	2	4	6	8	9	9
---	---	---	---	---	---	---

8.2-2) Since the counting sort starts at the back of the array (lines 10-12) when it's inputting the storage of the A-array into the B-array, the c-array holds where each number should be placed in the B-array. Since the B-array is inserted order ~~as~~ as they are seen at the end of the array A the output array is in the same order as they do in the input array. So counting sort is stable.

8.3-1) RADIX SORT

COW	SEA	TAB	BAR
DOG	TEA	BAR	BIG
SEA	MOB	EAR	BOX
RUG	TAB	TAR	COW
ROW	DOG	SEA	DIG
MOB	RUG	TEA	DOG
BOX →	DIG →	DIG →	EAR
TAB	BIG	BIG	FOX
BAR	BAR	MOB	MOB
EAR	EAR	DOG	NOW
TAR	TAR	COW	ROW
DIG	COW	ROW	RUG
BIG	ROW	NOW	SEA
TEA	NOW	BOX	TAB
NOW	BOX	FOX	TAR
FOX	FOX	RUG	TEA

SORTED!

8.3-2)

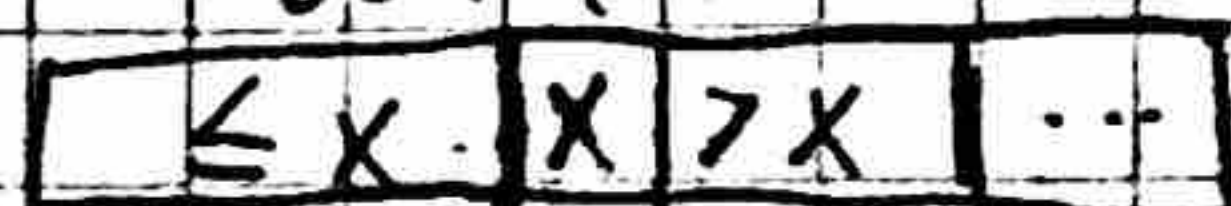
insertion sort
mergesort
heap sort
quicksort

Stable?

yes
yes
no
no

(Order is forgotten in binary tree)
(swaps non-adjacent elements)

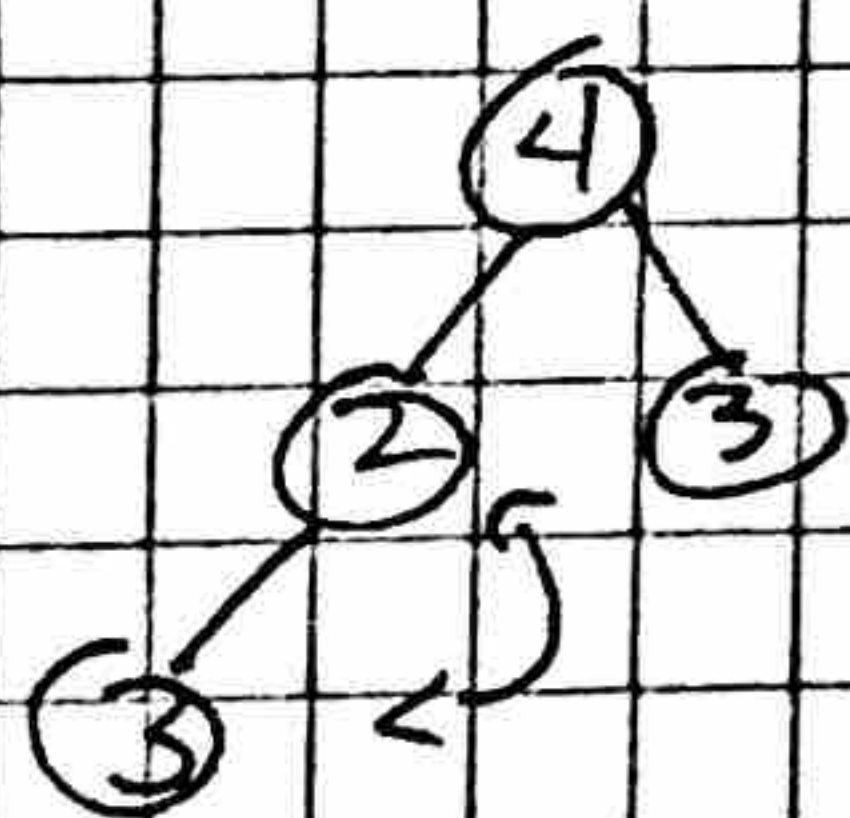
The insertion step in insertion sort



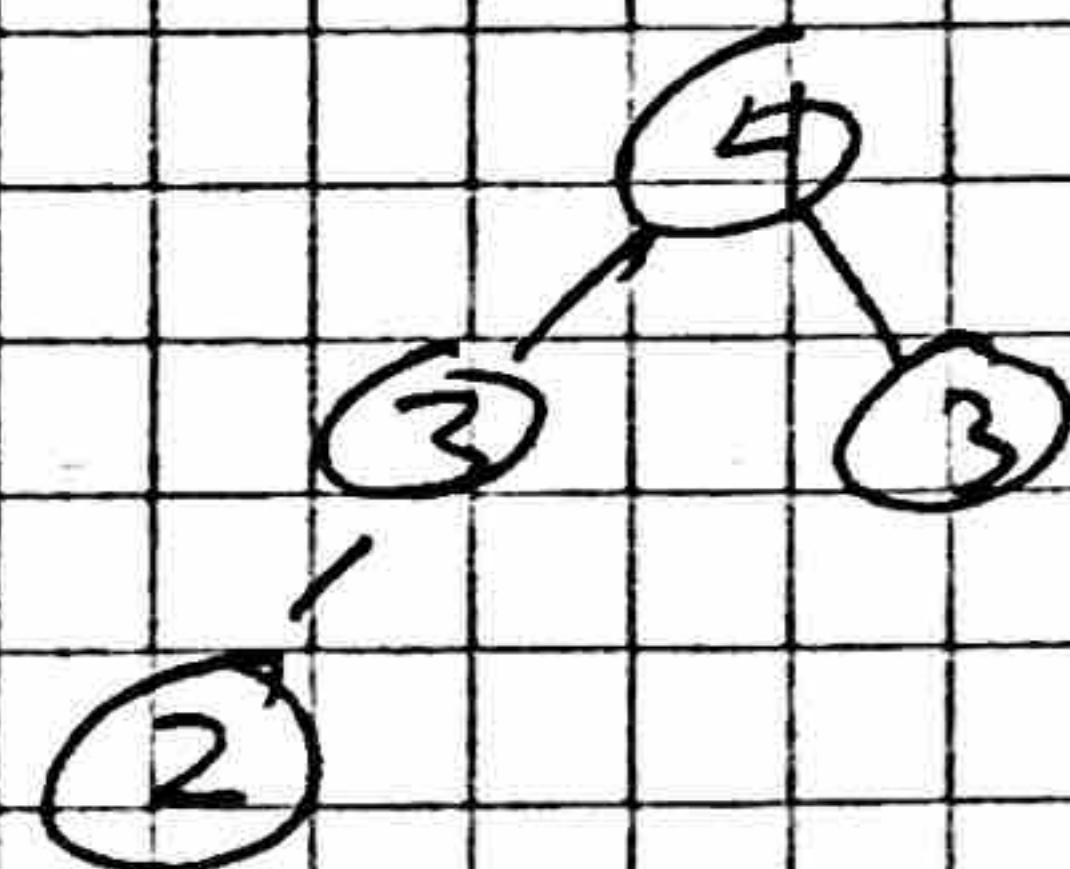
when x is placed into the sorted part of the array it stops in the location where all values are above x in the sorted portion and the ~~position~~ $\leq x$ ensures that if x is equal to the lower bound it won't go past that equal value. If it were $< x$ rather than $\leq x$ then insertion sort wouldn't be stable.

Merge sort is stable when implemented correctly. The Merge step is when data is being moved around in the sort. If the left and right arrays being compared have the same value, then in order for merge sort to be stable it must take the data from the left array before it takes it from the right array.

Heapsort is not a stable sort. There is the list $(4, 2, 3, 3)$

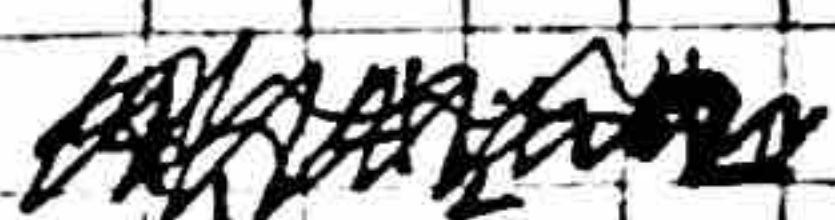
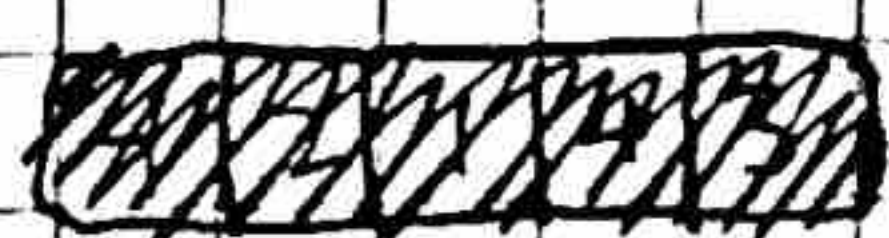


when maxheapify gets called the 2 and 3 switch



Max heapify switches the last 3 and 2 and that breaks stability $(4, 3, 3, 2)$

Quicksort is not stable. ~~It is not stable.~~



Consider list $(4_1, 4_2, 2)$

during the partition step

4_1 and 2 will get swapped

causing for the order to be

$(2, 4_2, 4_1)$ breaking stability.

8.3-3) Radix Sort (A, d)

for $i = 1$ to d

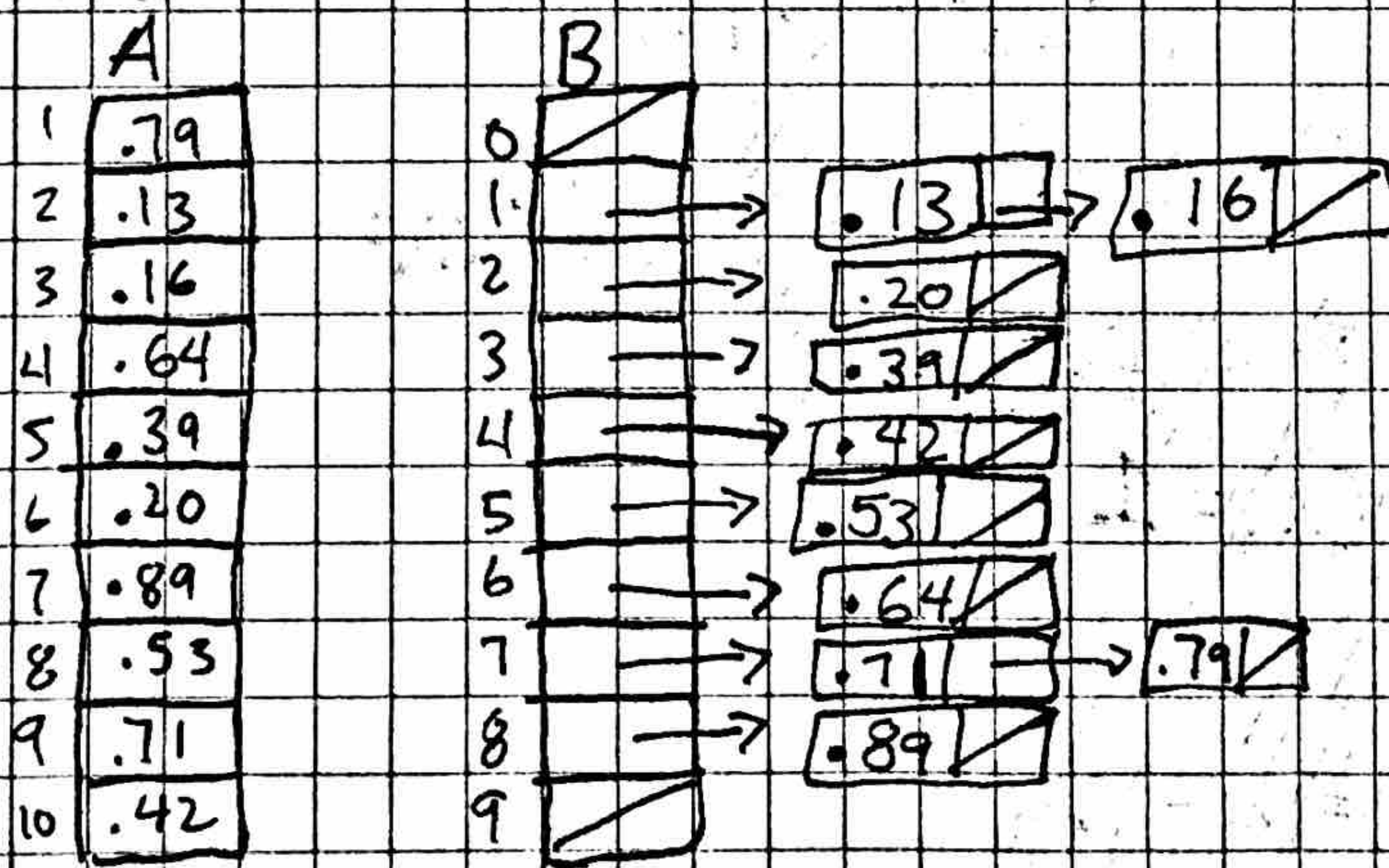
use a stable sort to sort array A on digit i

Initialization: The for loop starts with the least significant digit and uses a stable sort to sort that array on that digit i .

Maintenance: As the loop sorts based on the ones place and then sorts based on the tens place. If our sort method is not stable then the data won't be properly in the right order during each iteration through the loop. By ensuring the sort is stable we know we properly sort the ones place then the tens place all the way up until the d^{th} -digit.

Termination: The loop terminates ~~when~~ when $i = d+1$. So, we know that the invariant holds and the numbers are sorted based on d -digits.

8.4-1) $A = (.79, .13, .16, .64, .39, .20, .89, .53, .71, .42)$

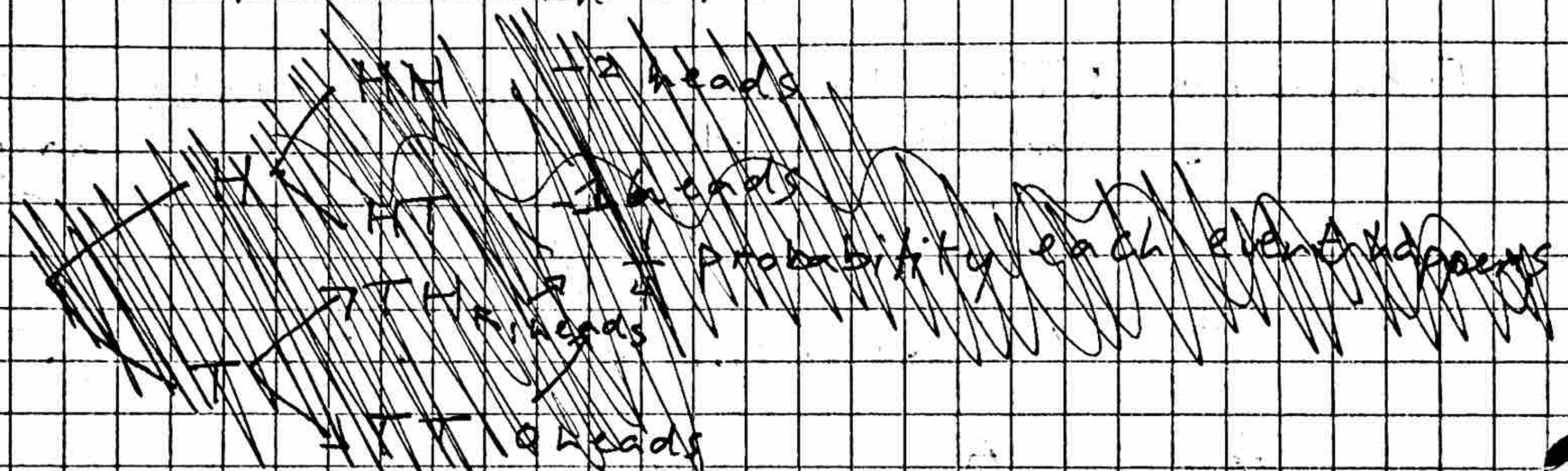


8.4-2) The worst case running time for bucket sort is $\Theta(n^2)$.

The case this happens is when the input goes into only one bucket and the data is in reverse order. This is because insertion sort runs in $\Theta(n^2)$ time.

Since we know our ~~input~~ input data ~~can~~ can have any formation of numbers between $[0, 1)$ then if we use merge sort instead of insertion sort in line 8 then the worst case run time would be $O(n \lg n)$.

~~when input comes in reverse order~~
~~when input is in reverse order~~



~~1/2 heads~~
~~1/4 heads~~
~~1/8 heads~~
~~1/16 heads~~
~~1/32 heads~~
~~1/64 heads~~
~~1/128 heads~~
~~1/256 heads~~
~~1/512 heads~~
~~1/1024 heads~~
~~1/2048 heads~~
~~1/4096 heads~~
~~1/8192 heads~~
~~1/16384 heads~~
~~1/32768 heads~~
~~1/65536 heads~~
~~1/131072 heads~~
~~1/262144 heads~~
~~1/524288 heads~~
~~1/1048576 heads~~
~~1/2097152 heads~~
~~1/4194304 heads~~
~~1/8388608 heads~~
~~1/16777216 heads~~
~~1/33554432 heads~~
~~1/67108864 heads~~
~~1/134217728 heads~~
~~1/268435456 heads~~
~~1/536870912 heads~~
~~1/1073741824 heads~~
~~1/2147483648 heads~~
~~1/4294967296 heads~~
~~1/8589934592 heads~~
~~1/17179869184 heads~~
~~1/34359738368 heads~~
~~1/68719476736 heads~~
~~1/137438953472 heads~~
~~1/274877906944 heads~~
~~1/549755813888 heads~~
~~1/1099511627776 heads~~
~~1/2199023255552 heads~~
~~1/4398046511104 heads~~
~~1/8796093022208 heads~~
~~1/17592186044416 heads~~
~~1/35184372088832 heads~~
~~1/70368744177664 heads~~
~~1/140737488355328 heads~~
~~1/281474976710656 heads~~
~~1/562949953421312 heads~~
~~1/1125899906842624 heads~~
~~1/2251799813685248 heads~~
~~1/4503599627370496 heads~~
~~1/9007199254740992 heads~~
~~1/18014398509481984 heads~~
~~1/36028797018963968 heads~~
~~1/72057594037927936 heads~~
~~1/144115188075855872 heads~~
~~1/288230376151711744 heads~~
~~1/576460752303423488 heads~~
~~1/1152921504606846976 heads~~
~~1/2305843009213693952 heads~~
~~1/4611686018427387904 heads~~
~~1/9223372036854775808 heads~~
~~1/18446744073709551616 heads~~
~~1/36893488147419103232 heads~~
~~1/73786976294838206464 heads~~
~~1/147573952589676412928 heads~~
~~1/295147905179352825856 heads~~
~~1/590295810358705651712 heads~~
~~1/1180591620717411303424 heads~~
~~1/2361183241434822606848 heads~~
~~1/4722366482869645213696 heads~~
~~1/9444732965739290427392 heads~~
~~1/18889465931478580854784 heads~~
~~1/37778931862957161709568 heads~~
~~1/75557863725914323419136 heads~~
~~1/151115727451828646838272 heads~~
~~1/302231454903657293676544 heads~~
~~1/604462909807314587353088 heads~~
~~1/1208925819614629174706176 heads~~
~~1/2417851639229258349412352 heads~~
~~1/4835703278458516698824704 heads~~
~~1/9671406556917033397649408 heads~~
~~1/19342813113834066795298816 heads~~
~~1/38685626227668133590597632 heads~~
~~1/77371252455336267181195264 heads~~
~~1/154742504910672534362390528 heads~~
~~1/309485009821345068724781056 heads~~
~~1/618970019642690137449562112 heads~~
~~1/1237940039285380274899124224 heads~~
~~1/2475880078570760549798248448 heads~~
~~1/4951760157141521099596496896 heads~~
~~1/9903520314283042199192993792 heads~~
~~1/19807040628566084398385987584 heads~~
~~1/39614081257132168796771975168 heads~~
~~1/79228162514264337593543950336 heads~~
~~1/158456325028528675187087900672 heads~~
~~1/316912650057057350374175801344 heads~~
~~1/633825300114114700748351602688 heads~~
~~1/1267650600228229401496703205376 heads~~
~~1/2535301200456458802993406410752 heads~~
~~1/5070602400912917605986812821504 heads~~
~~1/10141204801825835211973625643008 heads~~
~~1/20282409603651670423947251286016 heads~~
~~1/40564819207303340847894502572032 heads~~
~~1/81129638414606681695789005144064 heads~~
~~1/162259276829213363391578010288128 heads~~
~~1/324518553658426726783156020576256 heads~~
~~1/649037107316853453566312041152512 heads~~
~~1/1298074214633706907132624082305024 heads~~
~~1/2596148429267413814265248164610048 heads~~
~~1/5192296858534827628530496329220096 heads~~
~~1/10384593717069655257060992658440192 heads~~
~~1/20769187434139310514121985316880384 heads~~
~~1/41538374868278621028243970633760768 heads~~
~~1/83076749736557242056487941267521536 heads~~
~~1/166153499473114484112975882535043072 heads~~
~~1/332306998946228968225951765070086144 heads~~
~~1/664613997892457936451903530140172288 heads~~
~~1/1329227995784915872903807060280344576 heads~~
~~1/2658455991569831745807614120560689152 heads~~
~~1/5316911983139663491615228241121378304 heads~~
~~1/10633823966279326983230456482242756608 heads~~
~~1/21267647932558653966460912964485513216 heads~~
~~1/42535295865117307932921825928971026432 heads~~
~~1/85070591730234615865843651857942052864 heads~~
~~1/170141183460469231731687303715884105728 heads~~
~~1/340282366920938463463374607431768211456 heads~~
~~1/680564733841876926926749214863536422912 heads~~
~~1/1361129467683753853853498429727072845824 heads~~
~~1/2722258935367507707706996859454145691648 heads~~
~~1/5444517870735015415413993718908291383296 heads~~
~~1/10889035741470030830827987437816582766592 heads~~
~~1/21778071482940061661655974875633165533184 heads~~
~~1/43556142965880123323311949751266331066368 heads~~
~~1/87112285931760246646623899502532662132736 heads~~
~~1/174224571863520493293247799005065324265472 heads~~
~~1/348449143727040986586495598010130648530944 heads~~
~~1/696898287454081973172991196020261297061888 heads~~
~~1/1393796574908163946345982392040522594123776 heads~~
~~1/2787593149816327892691964784081045188247552 heads~~
~~1/5575186299632655785383929568162090376495104 heads~~
~~1/11150372599265311570767859136324180752990208 heads~~
~~1/22300745198530623141535718272648361505980416 heads~~
~~1/44601490397061246283071436545296723011960832 heads~~
~~1/89202980794122492566142873090593446023921664 heads~~
~~1/178405961588244985132285746181186892047843328 heads~~
~~1/356811923176489970264571492362373784095686656 heads~~
~~1/713623846352979940529142984724747568191373312 heads~~
~~1/1427247692705959881058285969449495136382746624 heads~~
~~1/2854495385411919762116571938898990272765493248 heads~~
~~1/5708990770823839524233143877797980545530986496 heads~~
~~1/11417981541647679048466287755595961091061972992 heads~~
~~1/22835963083295358096932575511191922182123945984 heads~~
~~1/45671926166590716193865151022383844364247891968 heads~~
~~1/91343852333181432387730302044767688728495783936 heads~~
~~1/182687704666362864775460604089535377456991567872 heads~~
~~1/365375409332725729550921208179070754913983135744 heads~~
~~1/730750818665451459101842416358141509827966271488 heads~~
~~1/1461501637330902918203684832716283019655932542976 heads~~
~~1/2923003274661805836407369665432566039311865085952 heads~~
~~1/5846006549323611672814739330865132078623730171904 heads~~
~~1/11692013098647223345629478661730264157247460343808 heads~~
~~1/23384026197294446691258957323460528314494920687616 heads~~
~~1/46768052394588893382517914646921056628989841375232 heads~~
~~1/93536104789177786765035829293842113257979682750464 heads~~
~~1/187072209578355573530071658587684226515959365500928 heads~~
~~1/374144419156711147060143317175368453031918731001856 heads~~
~~1/748288838313422294120286634350736906063837462003712 heads~~
~~1/1496577676626844588240573268701473812127674924007424 heads~~
~~1/2993155353253689176481146537402947624255349848014848 heads~~
~~1/5986310706507378352962293074805895248510699696029696 heads~~
~~1/11972621413014756705924586149611790497021399392059392 heads~~
~~1/23945242826029513411849172299223580994042798784118784 heads~~
~~1/47890485652059026823698344598447161988085597568237568 heads~~
~~1/95780971304118053647396689196894323976171195136475136 heads~~
~~1/191561942608236107294793378393788647952342390272950272 heads~~
~~1/383123885216472214589586756787577295904684780545900544 heads~~
~~1/766247770432944429179173513575154591809369561091801088 heads~~
~~1/1532495540865888858358347027150309183618739122183602176 heads~~
~~1/3064991081731777716716694054300618367237478244367204352 heads~~
~~1/6129982163463555433433388108601236734474956488734408704 heads~~
~~1/12259964326927110866866776217202473468949912977468817408 heads~~
~~1/24519928653854221733733552434404946937899825954937634816 heads~~
~~1/49039857307708443467467104868809893875799651909875269632 heads~~
~~1/98079714615416886934934209737619787751599303819750539264 heads~~
~~1/196159429230833773869868419475239575503198607639501078528 heads~~
~~1/392318858461667547739736838950479151006397215279002157056 heads~~
~~1/784637716923335095479473677900958302012794430558004314112 heads~~
~~1/1569275433846670190958947355801916604025588861116008628224 heads~~
~~1/3138550867693340381917894711603833208051177722232017256448 heads~~
~~1/6277101735386680763835789423207666416102355444464034512896 heads~~
~~1/12554203470773361527671578846415332832204710888928069025792 heads~~
~~1/25108406941546723055343157692830665664409421777856138051584 heads~~
~~1/50216813883093446110686315385661331328818843555712276103168 heads~~
~~1/100433627766186892221372630771322662657637687111424552206336 heads~~
~~1/200867255532373784442745261542645325315275374222849104412672 heads~~
~~1/401734511064747568885490523085290650630550748445698208825344 heads~~
~~1/803469022129495137770981046170581301261101496891396417650688 heads~~
~~1/1606938044258990275541962092341162602522202993782792835301376 heads~~
~~1/3213876088517980551083924184682325205044405987565585670602752 heads~~
~~1/6427752177035961102167848369364650410088811975131171341205504 heads~~
~~1/12855504354071922204335696738729300820177623950262342682411008 heads~~
~~1/25711008708143844408671393477458601640355247900524685364822016 heads~~
~~1/51422017416287688817342786954917203280710495801049370729644032 heads~~
~~1/102844034832575377634685573909834406561420991602098741459288064 heads~~
~~1/205688069665150755269371147819668813122841983204197482918576128 heads~~
~~1/411376139330301510538742295639337626245683966408394965837152256 heads~~
~~1/822752278660603021077484591278675252491367932816789931674304512 heads~~
~~1/1645504557321206042154969182557350504982735865633579863348609024 heads~~
~~1/3291009114642412084309938365114701009965471731267159726697218048 heads~~
~~1/6582018229284824168619876730229402019930943462534319453394436096 heads~~
~~1/13164036458569648337239753460458804039861886925068638906788872192 heads~~
~~1/26328072917139296674479506920917608079723773850137277813577744384 heads~~
~~1/52656145834278593348959013841835216159447547700274555627155488768 heads~~
~~1/105312291668557186697918027683670432318895095400549111254310977536 heads~~
~~1/210624583337114373395836055367340864637790190801098222508621955072 heads~~
~~1/421249166674228746791672110734681729275580381602196445017243910144 heads~~
~~1/842498333348457493583344221469363458551160763204392890034487820288 heads~~
~~1/1684996666696914987166688442938726917102321526408785780068975640576 heads~~
~~1/3369993333393829974333376885877453834204643052817571560137951281152 heads~~
~~1/6739986666787659948666753771754907668409286105635143120275902562304 heads~~
~~1/13479973333575319897333507543509815336818572211270286240551805124608 heads~~
~~1/26959946667150639794667015087019630673637144422540572481103610249216 heads~~
~~1/53919893334301279589334030174039261347274288845081144962207220498432 heads~~
~~1/107839786668602559178668060348078522694548577690162289924414440996864 heads~~
~~1/215679573337205118357336120696157045389097155380324579848828881993728 heads~~
~~1/431359146674410236714672241392314090778194310760649159697657763987456 heads~~
~~1/862718293348820473429344482784628181556388621521298319395315527974912 heads~~
~~1/1725436586697640946858688965569256363112777243042596638790631055949824 heads~~
~~1/3450873173395281893717377931138512726225554486085193277581262111899648 heads~~
~~1/6901746346790563787434755862277025452451108972170386555162524223799296 heads~~
~~1/13803492693581127574869511724554050904902217944340773110325048447598592 heads~~
~~1/27606985387162255149739023449108101809804435888681546220650096895197184 heads~~
~~1/55213970774324510299478046898216203619608871777363092441300193790394368 heads~~
~~1/110427941548649020598956093796432407239217743554726184882600387580788736 heads~~
~~1/220855883097298041197912187592864814478435487109452369765200775161577472 heads~~
~~1/441711766194596082395824375185729628956870974218904739530401550323154944 heads~~
~~1/883423532389192164791648750371459257913741948437809479060803100646309888 heads~~
~~1/1766847064778384329583297500742918515827483896875618958121606201292619776 heads~~
~~1/3533694129556768659166595001485837031654967793751237916243212402585239552 heads~~
~~1/7067388259113537318333190002971674063309935587502475832486424805170479104 heads~~
~~1/14134776518227074636666380005943348126619871175004951664972849610340958208 heads~~
~~1/28269553036454149273332760011886696253239742350009903329945699220681916416 heads~~
~~1/56539106072908298546665520023773392506479484700019806659891398441363832832 heads~~
~~1/113078212145816597093331040047546785012958969400039613319782796882727665664 heads~~
~~1/226156424291633194186662080095093570025917938800079226639565593765455331328 heads~~
~~1/452312848583266388373324160190187140051835877600158453279131187530910662656 heads~~
~~1/904625697166532776746648320380374280103671755200316906558262375061821325312 heads</~~

$$8.4-3) \quad P(X=0) = \frac{1}{4} \quad TT$$

$$P(X=1) = \frac{1}{2} \quad HT, TH$$

$$P(X=2) = \frac{1}{4} \quad HH$$

$$E[X] = 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} = \frac{1}{2} + \frac{1}{2} = 1$$

$$E[X^2] = 0^2 \cdot \frac{1}{4} + 1^2 \cdot \frac{1}{2} + 2^2 \cdot \frac{1}{4} = \frac{1}{2} + 2 = \boxed{\frac{3}{2}}$$

$$E^2[X] = 1^2 = \boxed{1}$$

8-4) a) deterministic algorithm would be just to compare each red pair with each blue pair. This results in $\Theta(n^2)$ comparisons

b) If we put each set of jugs into a decision tree then there are $n!$ permutations that map red jugs to blue jugs. for the length of the tree l we get the inequality

$$n! \leq l \leq 2^l$$

$$\Omega(n \lg n) \leq \lg(n!) \leq l$$

c) We will use a similar approach to randomized quicksort

- 1) ~~randomly~~ randomly choose a red pivot $O(1)$
- 2) find the matching blue pivot $O(n)$
- 3) Partition the blue jug around the red pivot and put the blue pivot in place $O(n)$
- 4) Partition the red jug around the blue pivot and put the ~~red~~ red pivot in place $O(n)$

The expected time of this algorithm is $O(n \lg n)$ because the partition is expected to happen somewhere near the middle of the array and that means there will be ~~roughly~~ $O(n \lg n)$ comparisons.

The worst number of comparisons is $\Theta(n^2)$ because we could get a bad partition that reduces the size to $n-1$ each iteration.