

Starting with this challenge, one of the most subtle changes is the following:

Previous challenge:

```
if (s.substring(startPos,(startPos+p.length())).indexOf(sb.toString().charAt(pos))!=-1)
```

This Challenge:

```
if (s.substring(startPos,(startPos+p.length())).indexOf(sb.toString().charAt(pos))!=0)
```

It has to be realised that in previous challenge we were not concerned if characters in String p appeared in any order in the block of String s.

In this new challenge, we require the match to be at index 0.

The reason is since the challenge required all permutations to be explored.

If we were to not make the above change, we would expect that ALL permutations to either appear in String s OR none of the permutations to appear in String s.

This is not what the challenge is trying to achieve!

```

185
186         for (int i=startPos; i<s.length();i++)
187     {
188         if(!sb.toString().isEmpty())
189     {
190         System.out.println("\nThis is STARTPOS: " + startPos);
191         System.out.println("This is sb length: " + sb.length());
192     }
193
194         for (int pos=0; pos<=sb.length(); pos++)
195     {
196         //if (hasCharFound)

```

We also need to remove this since it also contradicts the permutation. Instead we need to increment the pos variable.

```

231         pos++;
232
233         if (pos==p.length())
234     {
235         break;
236     }
237     //}
238 }

```

And break out the outer for loop when pos>=p.length()

Also need to modify this section since we are no longer interested in substring of String s..

```

//now the focus is on checking to see if the character of String p appears in that order in block size in String s
if (Character.toString(s.charAt(i)).indexOf(sb.toString().charAt(pos))==0)

```

Whilst I am continuing with this challenge, the first real issue occurred here.

```
194
195 //for (int pos=0; pos<=sb.length(); pos++)
196 //{
197     if (hasCharFound)
198     {
199         pos=0;
200     }
201     if (!sb.toString().isEmpty())
202     {
203         System.out.println("value of pos: " + pos);
204         System.out.println("value of sb: " + sb.toString());
205         System.out.println("Checking character: " + sb.toString().charAt(pos) +
206             " against the main String index: " + i + "(" + s.charAt(i) + ")");
207
208         System.out.println("SUBSTRING EXAMINED: " + s.substring(startPos, (startPos+p.length())));
209
210         //now the focus is on checking to see if the character of String p appears in that order in block size in String s
211         if (Character.toString(s.charAt(i)).indexOf(sb.toString().charAt(pos))!=0)
212         {
```

I am finding this error something I did not expect

I have not changed anything in my code to

```
*****RESTORING BACKUP OF STRINGBUILDER (String p): abc
This is STARTPOS: 7
This is sb length: 3
value of pos: 0
value of sb: abc
Checking character: a against the main String index: 7(a) TO index: 9(d)
SUBSTRING EXAMINED: acd
char found: a at index: 0
a has been removed from StringBuilder (String p)= abc
This is current StringBuilder (String p): bc
value of pos: 0
value of sb: bc
Checking character: b against the main String index: 7(a) TO index: 9(d)
SUBSTRING EXAMINED: acd
NO MATCH FOUND
StringBuilder being emptied: bc
*****RESTORING BACKUP OF STRINGBUILDER (String p): abc
```

Ln: 208, Col: 118

Run Share Command Line Arguments

```
This is STARTPOS: 8
This is sb length: 3
value of pos: 0
value of sb: bcb
Checking character: b against the main String index: 8(c)
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin 8, end 11, length 10
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:3319)
    at java.base/java.lang.String.substring(String.java:1874)
    at Main.findAnagrams(Main.java:208)
    at Main.main(Main.java:52)
```

If we examine the previous challenge it had the intelligence to stop at 7 when processing the same test condition

```
15 //TEST CASE 1:
16 findAnagrams("cbaebabacd", "abc");
17
```

Strictly speaking, using the do while loop in my first code, it should have also failed to progress since I have not modified these...
But we know the structure is technically correct:

```
}while(p.length()+startPos<s.length());
```

We can see that this is post
test evaluation

We can clearly see that

p.length()=3

startPos= 7

s.length= 10

```
NO MATCH FOUND
StringBuilder being emptied: ba
*****RESTORING BACKUP OF STRINGBUILDER (String p): bba

This is STARTPOS: 7
This is sb length: 3
This is s length: 10
value of pos: 0
value of sb: bba
Checking character: b against the main String index: 7(a)
SUBSTRING EXAMINED: acd
NO MATCH FOUND
StringBuilder being emptied: bba
*****RESTORING BACKUP OF STRINGBUILDER (String p): bba
aaa Subset: 2 at cycle number: 6
THIS IS SB: aaa

This is STARTPOS: 8
This is sb length: 3
This is s length: 10
value of pos: 0
```

There is absolutely no
reason for the code to
reach here...

I will keep this code in my
repository.

Perhaps I will opt for a
while loop instead

```
178
179         while(p.length()+startPos<s.length())
180     {
181         do
182         {
183             startPos=counter;
184
185             for (int i=startPos; i<s.length();i++)
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Ln: 174, Col: 69

Run Share Command Line Arguments

```
This is STARTPOS: 7
This is sb length: 3
This is s length: 10
value of pos: 0
value of sb: bba
Checking character: b against the main String index: 7(a)
SUBSTRING EXAMINED: acd
NO MATCH FOUND
StringBuilder being emptied: bba
*****RESTORING BACKUP OF STRINGBUILDER (String p): bba
abb Subset: 2 at cycle number: 7
THIS IS SB: abb
aba Subset: 3 at cycle number: 7
THIS IS SB: aba
acb Subset: 4 at cycle number: 7
THIS IS SB: acb
cba Subset: 5 at cycle number: 7
THIS IS SB: cba
aca Subset: 6 at cycle number: 7
THIS IS SB: aca

** Process exited - Return Code: 0 **
```

My next aim is to determine why it has not processed other scenarios in valuesSet of ALL permutations and just skipped them!

The program has terminated properly

In regards to investigating the next permutation, I would need to simply look at the area of code in question...

Logic suggests it has to be related to the following condition.

```
while(p.length()+startPos<s.length())
```

We know from the previous challenge, once it traversed through String s, there was no need to start the process again from the offset.

We know with this challenge, we still have outstanding permutations...

So logic suggests that once it exits out of the inner do while loop, we would need to set startPos back to 0 and also counter to 0

I have completed the following change:

```
259     } //end of while loop while(p.length()+startPos<s.length())
260
261     | //but we will find that counter is still one greater than startPos in which the while loop terminated
262     | // so effectively, startPos will once again become 8.
263     | //and for a main String s where the length is 10 characters, this will invalidate the situation and generate a
264     | //StringIndexOutOfBoundsException
265     | //effectively I have set the counter =0 once it exits the while loop.
266     | //we know at this point it is in the for loop where it is processing the permutations.
267     | //I believe all the conditions are now set for a fresh execution....
268
269     counter=0;
270     startPos=0;
271
```

And finally it runs through all my test executions...

Now clearly, it would be best that I compose my test case String with all the permutations...

However as I reached the end of the challenge, I had realised that I have made a fundamental mistake during the process of getting the permutations. I had clearly forgotten to remove the selection.

Infact my selection was still taking place via random index from a String. So the only route forward was to configure a List with characters from String p. And attempt selection.

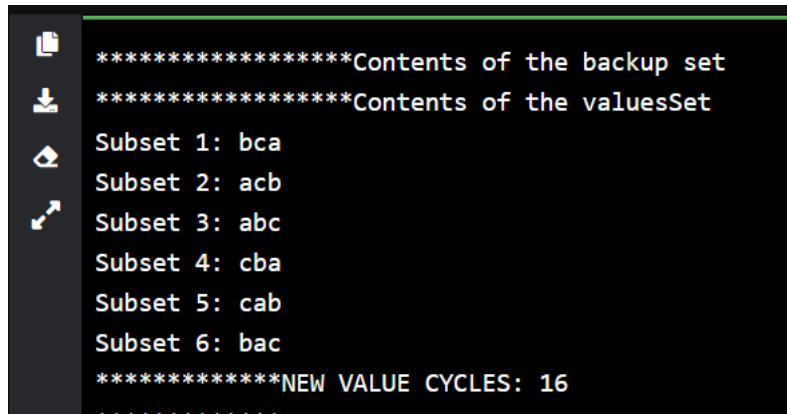
```
*****Contents of the backup set
*****Contents of the valuesSet
Subset 1: abb
Subset 2: acb
Subset 3: cbb
Subset 4: bcc
Subset 5: bbb
Subset 6: bac
*****NEW VALUE CYCLES: 6
*****RUNNING TOTAL CYCLES: 6
***PROCESSING SET AT INDEX: 0
```

THIS IS EQUIVALENT TO
PERMUTATION WITHOUT
REPLACEMENT AND IT
WAS NOT THE NATURE
OF THE CHALLENGE
SINCE ALL LETTERS HAD
TO BE USED EXACTLY
ONCE AND DISCARDED
AGAIN FROM SELECTION

Instead of detailing walkthrough, I have annotated my code and quite efficiently applied necessary changes.

It once again caused confusion as to use Lists and how to perform random selection without replication of indexes. I eventually reached a solution.

And it can be seen that selection is now as expected...

A screenshot of a terminal window with a dark background and light green text. On the left side, there is a vertical toolbar with icons for file operations (copy, paste, download, upload, home, search) and a window management icon. The terminal output consists of several lines of text, some enclosed in asterisks. The text reads: "*****Contents of the backup set", "*****Contents of the valuesSet", "Subset 1: bca", "Subset 2: acb", "Subset 3: abc", "Subset 4: cba", "Subset 5: cab", "Subset 6: bac", and "*****NEW VALUE CYCLES: 16".

```
*****Contents of the backup set
*****Contents of the valuesSet
Subset 1: bca
Subset 2: acb
Subset 3: abc
Subset 4: cba
Subset 5: cab
Subset 6: bac
*****NEW VALUE CYCLES: 16
```

It will be difficult to present test cases, however I will perform a full output of all the screen messages...