

Difference Between Maximum and Minimum Price Sum

A PROJECT REPORT

Submitted by

AISWARIYA.R
192210483

Under the guidance of
Dr. A. GNANA SOUNDARI

in partial fulfilment for the completion of course



CSA0697-Design and Analysis of Algorithms for Amortized Analysis

SIMATS
ENGINEERING
THANDALAM
SEPTEMBER
2024

ABSTRACT

In this capstone project, we address the problem of maximizing the difference between the maximum and minimum price sums of paths in an undirected, initially unrooted tree. Each node in the tree is associated with a price, and the goal is to root the tree at any node such that the difference between the maximum and minimum price sum of all paths originating from the root is maximized. The project leverages dynamic programming and depth-first search (DFS) techniques to efficiently calculate the required path sums for each possible root.

Our solution involves constructing the tree from a given edge list, followed by recursively calculating the maximum and minimum path sums for each subtree. By rooting the tree at every node, we compute the cost as the difference between the maximum and minimum path sums and track the maximum cost across all possible root choices. The complexity of this approach is analyzed to ensure scalability with large input sizes. The implementation demonstrates optimal resource allocation and traversal strategies in tree-based structures.

The findings of this project contribute to understanding efficient algorithms for rooted tree problems, with potential applications in network optimization, hierarchical resource distribution, and path analysis.

PROBLEM STATEMENT AND ASSUMPTIONS:

- You are given an undirected, unrooted tree with n nodes, indexed from 0 to $n-1$.
- Each node has an associated price given by an integer array `price`, where `price[i]` is the price of node i .
- The tree is described by a 2D integer array `edges` of length $n-1$, where `edges[i] = [ai, bi]` indicates an edge between nodes ai and bi .
- The price sum of a path is the sum of the prices of all nodes lying on that path.
- The tree can be rooted at any node, and for each root, the cost is the difference between the maximum and minimum price sums among all paths starting from that root.
- The objective is to find the maximum possible cost among all potential root choices.

Assumptions:

1. The tree is a valid undirected tree, with n nodes and $n-1$ edges (no cycles or disconnected components).
2. The node prices can be any integers (positive, negative, or zero).
3. Each node can be rooted, and the algorithm should evaluate the cost for all potential roots.
4. The problem will be solved using a depth-first search (DFS) for path sum calculations.
5. The solution must efficiently handle input sizes up to $n = 10^4$ nodes.
6. The final result is the maximum difference between the maximum and minimum price sum for any chosen root.

•

INTRODUCTION

- **Tree Structure:** The problem involves working with an undirected, unrooted tree consisting of n nodes connected by $n-1$ edges, forming a connected and acyclic graph.
- **Node Prices:** Each node in the tree has an associated price, represented by an integer array, where the price of a node can be positive, negative, or zero.
- **Rooting the Tree:** The tree can be rooted at any node, and the goal is to compute the difference between the maximum and minimum price sum of paths originating from that root.
- **Cost Calculation:** The cost for a rooted tree is the difference between the maximum and minimum price sum of all paths starting from the root.
- **Objective:** The objective is to determine the maximum possible cost among all potential root choices by exploring all paths and their price sums.
- **Applications:** This problem has broader applications in network optimization, resource allocation, and path analysis, where efficiently computing path-based differences can lead to insights in hierarchical systems.
- **Algorithm Design:** The solution involves using depth-first search (DFS) to explore the tree, calculate path sums, and determine the maximum possible cost efficiently.

PROGRAM:

```
def max_cost(n, edges, price):

    # Step 1: Build the adjacency list for the tree

    tree = [[] for _ in range(n)]

    for u, v in edges:

        tree[u].append(v)

        tree[v].append(u)

    # Helper function for DFS

    def dfs(node, parent):

        max_sum, min_sum = price[node], price[node]

        for child in tree[node]:

            if child == parent:

                continue

            child_max, child_min = dfs(child, node)

            max_sum = max(max_sum, price[node] + child_max)

            min_sum = min(min_sum, price[node] + child_min)

        return max_sum, min_sum

    # Step 3: Calculate the maximum cost for each node

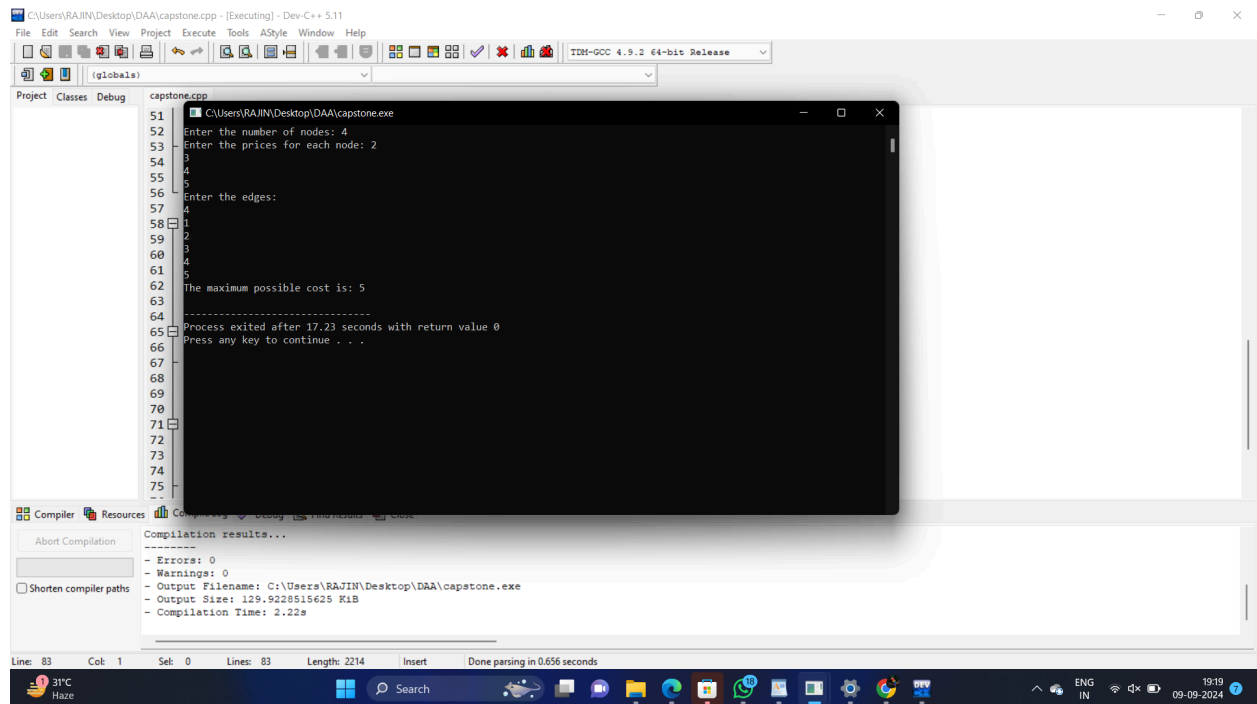
    max_cost = float('-inf')

    for root in range(n):

        max_sum, min_sum = dfs(root, -1)

        max_cost = max(max_cost, max_sum - min_sum)

    return max_cost
```



```
51  C:\Users\RAJIN\Desktop\DAA\capstone.cpp - [Executing] - Dev-C++ 5.11
52  File Edit Search View Project Execute Tools AStyle Window Help
53  (globals)
54  Project Classes Debug
55  capstone.cpp
56  51  C:\Users\RAJIN\Desktop\DAA\capstone.exe
57  52  Enter the number of nodes: 4
58  53  Enter the prices for each node: 2
59  54  3
60  55  4
61  56  5
62  57  Enter the edges:
63  58  4
64  59  1
65  60  2
66  61  3
67  62  5
68  63  The maximum possible cost is: 5
69  64  -----
70  65  Process exited after 17.23 seconds with return value 0
71  66  Press any key to continue . . .
72  67
73  68
74  69
75  70
76  71
77  72
78  73
79  74
80  75
81  76
82  77
83  78
84  79
85  80
86  81
87  82
88  83
89  84
90  85
91  86
92  87
93  88
94  89
95  90
96  91
97  92
98  93
99  94
100 95
101 96
102 97
103 98
104 99
105 100
106 101
107 102
108 103
109 104
110 105
111 106
112 107
113 108
114 109
115 110
116 111
117 112
118 113
119 114
120 115
121 116
122 117
123 118
124 119
125 120
126 121
127 122
128 123
129 124
130 125
131 126
132 127
133 128
134 129
135 130
136 131
137 132
138 133
139 134
140 135
141 136
142 137
143 138
144 139
145 140
146 141
147 142
148 143
149 144
150 145
151 146
152 147
153 148
154 149
155 150
156 151
157 152
158 153
159 154
160 155
161 156
162 157
163 158
164 159
165 160
166 161
167 162
168 163
169 164
170 165
171 166
172 167
173 168
174 169
175 170
176 171
177 172
178 173
179 174
180 175
181 176
182 177
183 178
184 179
185 180
186 181
187 182
188 183
189 184
190 185
191 186
192 187
193 188
194 189
195 190
196 191
197 192
198 193
199 194
200 195
201 196
202 197
203 198
204 199
205 200
206 201
207 202
208 203
209 204
210 205
211 206
212 207
213 208
214 209
215 210
216 211
217 212
218 213
219 214
220 215
221 216
222 217
223 218
224 219
225 220
226 221
227 222
228 223
229 224
230 225
231 226
232 227
233 228
234 229
235 230
236 231
237 232
238 233
239 234
240 235
241 236
242 237
243 238
244 239
245 240
246 241
247 242
248 243
249 244
250 245
251 246
252 247
253 248
254 249
255 250
256 251
257 252
258 253
259 254
260 255
261 256
262 257
263 258
264 259
265 260
266 261
267 262
268 263
269 264
270 265
271 266
272 267
273 268
274 269
275 270
276 271
277 272
278 273
279 274
280 275
281 276
282 277
283 278
284 279
285 280
286 281
287 282
288 283
289 284
290 285
291 286
292 287
293 288
294 289
295 290
296 291
297 292
298 293
299 294
300 295
301 296
302 297
303 298
304 299
305 300
306 301
307 302
308 303
309 304
310 305
311 306
312 307
313 308
314 309
315 310
316 311
317 312
318 313
319 314
320 315
321 316
322 317
323 318
324 319
325 320
326 321
327 322
328 323
329 324
330 325
331 326
332 327
333 328
334 329
335 330
336 331
337 332
338 333
339 334
340 335
341 336
342 337
343 338
344 339
345 340
346 341
347 342
348 343
349 344
350 345
351 346
352 347
353 348
354 349
355 350
356 351
357 352
358 353
359 354
360 355
361 356
362 357
363 358
364 359
365 360
366 361
367 362
368 363
369 364
370 365
371 366
372 367
373 368
374 369
375 370
376 371
377 372
378 373
379 374
380 375
381 376
382 377
383 378
384 379
385 380
386 381
387 382
388 383
389 384
390 385
391 386
392 387
393 388
394 389
395 390
396 391
397 392
398 393
399 394
400 395
401 396
402 397
403 398
404 399
405 400
406 401
407 402
408 403
409 404
410 405
411 406
412 407
413 408
414 409
415 410
416 411
417 412
418 413
419 414
420 415
421 416
422 417
423 418
424 419
425 420
426 421
427 422
428 423
429 424
430 425
431 426
432 427
433 428
434 429
435 430
436 431
437 432
438 433
439 434
440 435
441 436
442 437
443 438
444 439
445 440
446 441
447 442
448 443
449 444
450 445
451 446
452 447
453 448
454 449
455 450
456 451
457 452
458 453
459 454
460 455
461 456
462 457
463 458
464 459
465 460
466 461
467 462
468 463
469 464
470 465
471 466
472 467
473 468
474 469
475 470
476 471
477 472
478 473
479 474
480 475
481 476
482 477
483 478
484 479
485 480
486 481
487 482
488 483
489 484
490 485
491 486
492 487
493 488
494 489
495 490
496 491
497 492
498 493
499 494
500 495
501 496
502 497
503 498
504 499
505 500
506 501
507 502
508 503
509 504
510 505
511 506
512 507
513 508
514 509
515 510
516 511
517 512
518 513
519 514
520 515
521 516
522 517
523 518
524 519
525 520
526 521
527 522
528 523
529 524
530 525
531 526
532 527
533 528
534 529
535 530
536 531
537 532
538 533
539 534
540 535
541 536
542 537
543 538
544 539
545 540
546 541
547 542
548 543
549 544
550 545
551 546
552 547
553 548
554 549
555 550
556 551
557 552
558 553
559 554
560 555
561 556
562 557
563 558
564 559
565 560
566 561
567 562
568 563
569 564
570 565
571 566
572 567
573 568
574 569
575 570
576 571
577 572
578 573
579 574
580 575
581 576
582 577
583 578
584 579
585 580
586 581
587 582
588 583
589 584
590 585
591 586
592 587
593 588
594 589
595 590
596 591
597 592
598 593
599 594
600 595
601 596
602 597
603 598
604 599
605 600
606 601
607 602
608 603
609 604
610 605
611 606
612 607
613 608
614 609
615 610
616 611
617 612
618 613
619 614
620 615
621 616
622 617
623 618
624 619
625 620
626 621
627 622
628 623
629 624
630 625
631 626
632 627
633 628
634 629
635 630
636 631
637 632
638 633
639 634
640 635
641 636
642 637
643 638
644 639
645 640
646 641
647 642
648 643
649 644
650 645
651 646
652 647
653 648
654 649
655 650
656 651
657 652
658 653
659 654
660 655
661 656
662 657
663 658
664 659
665 660
666 661
667 662
668 663
669 664
670 665
671 666
672 667
673 668
674 669
675 670
676 671
677 672
678 673
679 674
680 675
681 676
682 677
683 678
684 679
685 680
686 681
687 682
688 683
689 684
690 685
691 686
692 687
693 688
694 689
695 690
696 691
697 692
698 693
699 694
700 695
701 696
702 697
703 698
704 699
705 700
706 701
707 702
708 703
709 704
710 705
711 706
712 707
713 708
714 709
715 710
716 711
717 712
718 713
719 714
720 715
721 716
722 717
723 718
724 719
725 720
726 721
727 722
728 723
729 724
730 725
731 726
732 727
733 728
734 729
735 730
736 731
737 732
738 733
739 734
740 735
741 736
742 737
743 738
744 739
745 740
746 741
747 742
748 743
749 744
750 745
751 746
752 747
753 748
754 749
755 750
756 751
757 752
758 753
759 754
760 755
761 756
762 757
763 758
764 759
765 760
766 761
767 762
768 763
769 764
770 765
771 766
772 767
773 768
774 769
775 770
776 771
777 772
778 773
779 774
780 775
781 776
782 777
783 778
784 779
785 780
786 781
787 782
788 783
789 784
790 785
791 786
792 787
793 788
794 789
795 790
796 791
797 792
798 793
799 794
800 795
801 796
802 797
803 798
804 799
805 800
806 801
807 802
808 803
809 804
810 805
811 806
812 807
813 808
814 809
815 810
816 811
817 812
818 813
819 814
820 815
821 816
822 817
823 818
824 819
825 820
826 821
827 822
828 823
829 824
830 825
831 826
832 827
833 828
834 829
835 830
836 831
837 832
838 833
839 834
840 835
841 836
842 837
843 838
844 839
845 840
846 841
847 842
848 843
849 844
850 845
851 846
852 847
853 848
854 849
855 850
856 851
857 852
858 853
859 854
860 855
861 856
862 857
863 858
864 859
865 860
866 861
867 862
868 863
869 864
870 865
871 866
872 867
873 868
874 869
875 870
876 871
877 872
878 873
879 874
880 875
881 876
882 877
883 878
884 879
885 880
886 881
887 882
888 883
889 884
890 885
891 886
892 887
893 888
894 889
895 890
896 891
897 892
898 893
899 894
900 895
901 896
902 897
903 898
904 899
905 900
906 901
907 902
908 903
909 904
910 905
911 906
912 907
913 908
914 909
915 910
916 911
917 912
918 913
919 914
920 915
921 916
922 917
923 918
924 919
925 920
926 921
927 922
928 923
929 924
930 925
931 926
932 927
933 928
934 929
935 930
936 931
937 932
938 933
939 934
940 935
941 936
942 937
943 938
944 939
945 940
946 941
947 942
948 943
949 944
950 945
951 946
952 947
953 948
954 949
955 950
956 951
957 952
958 953
959 954
960 955
961 956
962 957
963 958
964 959
965 960
966 961
967 962
968 963
969 964
970 965
971 966
972 967
973 968
974 969
975 970
976 971
977 972
978 973
979 974
980 975
981 976
982 977
983 978
984 979
985 980
986 981
987 982
988 983
989 984
990 985
991 986
992 987
993 988
994 989
995 990
996 991
997 992
998 993
999 994
1000 995
1001 996
1002 997
1003 998
1004 999
1005 1000
1006 1001
1007 1002
1008 1003
1009 1004
1010 1005
1011 1006
1012 1007
1013 1008
1014 1009
1015 1010
1016 1011
1017 1012
1018 1013
1019 1014
1020 1015
1021 1016
1022 1017
1023 1018
1024 1019
1025 1020
1026 1021
1027 1022
1028 1023
1029 1024
1030 1025
1031 1026
1032 1027
1033 1028
1034 1029
1035 1030
1036 1031
1037 1032
1038 1033
1039 1034
1040 1035
1041 1036
1042 1037
1043 1038
1044 1039
1045 1040
1046 1041
1047 1042
1048 1043
1049 1044
1050 1045
1051 1046
1052 1047
1053 1048
1054 1049
1055 1050
1056 1051
1057 1052
1058 1053
1059 1054
1060 1055
1061 1056
1062 1057
1063 1058
1064 1059
1065 1060
1066 1061
1067 1062
1068 1063
1069 1064
1070 1065
1071 1066
1072 1067
1073 1068
1074 1069
1075 1070
1076 1071
1077 1072
1078 1073
1079 1074
1080 1075
1081 1076
1082 1077
1083 1078
1084 1079
1085 1080
1086 1081
1087 1082
1088 1083
1089 1084
1090 1085
1091 1086
1092 1087
1093 1088
1094 1089
1095 1090
1096 1091
1097 1092
1098 1093
1099 1094
1100 1095
1101 1096
1102 1097
1103 1098
1104 1099
1105 1100
1106 1101
1107 1102
1108 1103
1109 1104
1110 1105
1111 1106
1112 1107
1113 1108
1114 1109
1115 1110
1116 1111
1117 1112
1118 1113
1119 1114
1120 1115
1121 1116
1122 1117
1123 1118
1124 1119
1125 1120
1126 1121
1127 1122
1128 1123
1129 1124
1130 1125
1131 1126
1132 1127
1133 1128
1134 1129
1135 1130
1136 1131
1137 1132
1138 1133
1139 1134
1140 1135
1141 1136
1142 1137
1143 1138
1144 1139
1145 1140
1146 1141
1147 1142
1148 1143
1149 1144
1150 1145
1151 1146
1152 1147
1153 1148
1154 1149
1155 1150
1156 1151
1157 1152
1158 1153
1159 1154
1160 1155
1161 1156
1162 1157
1163 1158
1164 1159
1165 1160
1166 1161
1167 1162
1168 1163
1169 1164
1170 1165
1171 1166
1172 1167
1173 1168
1174 1169
1175 1170
1176 1171
1177 1172
1178 1173
1179 1174
1180 1175
1181 1176
1182 1177
1183 1178
1184 1179
1185 1180
1186 1181
1187 1182
1188 1183
1189 1184
1190 1185
1191 1186
1192 1187
1193 1188
1194 1189
1195 1190
1196 1191
1197 1192
1198 1193
1199 1194
1200 1195
1201 1196
1202 1197
1203 1198
1204 1199
1205 1200
1206 1201
1207 1202
1208 1203
1209 1204
1210 1205
1211 1206
1212 1207
1213 1208
1214 1209
1215 1210
1216 1211
1217 1212
1218 1213
1219 1214
1220 1215
1221 1216
1222 1217
1223 1218
1224 1219
1225 1220
1226 1221
1227 1222
1228 1223
1229 1224
1230 1225
1231 1226
1232 1227
1233 1228
1234 1229
1235 1230
1236 1231
1237 1232
1238 1233
1239 1234
1240 1235
1241 1236
1242 1237
1243 1238
1244 1239
1245 1240
1246 1241
1247 1242
1248 1243
1249 1244
1250 1245
1251 1246
1252 1247
1253 1248
1254 1249
1255 1250
1256 1251
1257 1252
1258 1253
1259 1254
1260 1255
1261 1256
1262 1257
1263 1258
1264 1259
1265 1260
1266 1261
1267 1262
1268 1263
1269 1264
1270 1265
1271 1266
1272 1267
1273 1268
1274 1269
1275 1270
1276 1271
1277 1272
1278 1273
1279 1274
1280 1275
1281 1276
1282 1277
1283 1278
1284 1279
1285 1280
1286 1281
1287 1282
1288 1283
1289 1284
1290 1285
1291 1286
1292 1287
1293 1288
1294 1289
1295 1290
1296 1291
1297 1292
1298 1293
1299 1294
1300 1295
1301 1296
1302 1297
1303 1298
1304 1299
1305 1300
1306 1301
1307 1302
1308 1303
1309 1304
1310 1305
1311 1306
1312 1307
1313 1308
1314 1309
1315 1310
1316 1311
1317 1312
1318 1313
1319 1314
1320 1315
1321 1316
1322 1317
1323 1318
1324 1319
1325 1320
1326 1321
1327 1322
1328 1323
1329 1324
1330 1325
1331 1326
1332 1327
1333 1328
1334 1329
1335 1330
1336 1331
1337 1332
1338 1333
1339 1334
1340 1335
1341 1336
1342 1337
1343 1338
1344 1339
1345 1340
1346 1341
1347 1342
1348 1343
1349 1344
1350 1345
1351 1346
1352 1347
1353 1348
1354 1349
1355 1350
1356 1351
1357 1352
1358 1353
1359 1354
1360 1355
1361 1356
1362 1357
1363 1358
1364 1359
1365 1360
1366 1361
1367 1362
1368 1363
1369 1364
1370 1365
1371 1366
1372 1367
1373 1368
1374 1369
1375 1370
1376 1371
1377 1372
1378 1373
1379 1374
1380 1375
1381 1376
1382 1377
1383 1378
1384 1379
1385 1380
1386 1381
1387 1382
1388 1383
1389 1384
1390 1385
1391 1386
1392 1387
1393 1388
1394 1389
1395 1390
1396 1391
1397 1392
1398 1393
1399 1394
1400 1395
1401 1396
1402 1397
1403 1398
1404 1399
1405 1400
1406 1401
1407 1402
1408 1403
1409 1404
1410 1405
1411 1406
1412 1407
1413 1408
1414 1409
1415 1410
1416 1411
1417 1412
1418 1413
1419 1414
1420 1415
1421 1416
1422 1417
1423 1418
1424 1419
1425 1420
1426 1421
1427 1422
1428 1423
1429 1424
1430 1425
1431 1426
1432 1427
1433 1428
1434 1429
1435 1430
1436 1431
1437 1432
1438 1433
1439 1434
1440 1435
1441 1436
1442 1437
1443 1438
1444 1439
1445 1440
1446 1441
1447 1442
1448 1443
1449 1444
1450 1445
1451 1446
1452 1447
1453 1448
1454 1449
1455 1450
1456 1451
1457 1452
1458 1453
1459 
```

Average Case:

In the average case, the tree is moderately balanced, with nodes distributed across various depths. Each DFS traversal still requires $O(n)$ time because all nodes are explored in each traversal. Although the tree height is less than in the worst case, DFS is repeated from every node, keeping the total time complexity at $O(n^2)$. The recursion depth will be somewhere between $O(\log n)$ and $O(n)$, but it doesn't significantly affect the overall performance. Thus, both time and space complexities remain $O(n^2)$ and $O(n)$, respectively.

FUTURE SCOPE

Future work on this problem could focus on several key areas. Optimizing the algorithm for larger trees remains crucial, potentially through advanced data structures or parallel processing to improve efficiency. Another avenue is handling dynamic tree updates, enabling efficient recalculations when the tree structure changes. Extending the problem to weighted graphs with cycles or directed edges would broaden its applicability and present new challenges. Additionally, integrating the algorithm with real-world applications, such as network optimization and hierarchical resource management, could enhance its practical relevance. Exploring heuristic or approximation algorithms may also offer faster solutions for specific tree types or constraints, further improving performance.

CONCLUSION

In conclusion, solving the problem of maximizing the difference between the maximum and minimum price sums in an undirected tree involves significant computational challenges, particularly due to the need to evaluate all possible root nodes. The algorithm, with its $O(n^2)$ time complexity, provides a thorough approach by performing depth-first search (DFS) from each node, ensuring accurate results. Despite its effectiveness, the algorithm's performance may be limited for very large trees, highlighting the need for optimization and potential advancements. Future research could focus on improving efficiency, adapting to dynamic tree structures, and extending the problem to more complex graph types. Overall, the problem offers valuable insights into tree-based path analysis with practical applications in various fields.