Report: Lab 09

Submitted by: Aakash Banik (16i190010)

Question 0:

File: ex0.py

Question 1:

Subpart a:

File: ex1a.py

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1a.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684
/Lab 9')
Reloaded modules: net0
The sequence of nodes in the order they are visited are:  [1, 5, 3, 2, 4]
>>>
```

Figure 1: Output when Depth First Search was implemented on net0

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1a.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684
/Lab 9')
Reloaded modules: net0
The sequence of nodes in the order they are visited are:  [1, 2, 6, 9, 7, 5, 3, 10, 4, 8]
>>>
```

Figure 2: Output when Depth First Search was implemented on net1

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1a.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684
/Lab 9')
Reloaded modules: net1
The sequence of nodes in the order they are visited are:  [1, 29, 39, 13, 12, 49, 48, 3, 50, 7, 40, 11,
36, 15, 43, 26, 6, 31, 21, 2, 22, 37, 20, 16, 27, 9, 47, 44, 19, 34, 38, 32, 5, 10, 42, 30, 14, 17, 46,
33, 41, 35, 4, 23, 8, 18, 45, 28, 25, 24]
>>>
```

Figure 3: Output when Depth First Search was implemented on net2

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1a.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684
/Lab 9')
Reloaded modules: net2
The sequence of nodes in the order they are visited are:  [1, 75, 96, 34, 99, 74, 67, 92, 24, 11, 61, 81
, 41, 69, 59, 72, 23, 4, 86, 87, 2, 78, 73, 82, 90, 91, 76, 80, 36, 19, 42, 38, 46, 32, 30, 56, 58, 68,
6, 17, 51, 5, 70, 39, 43, 66, 54, 20, 48, 29, 95, 97, 57, 63, 55, 18, 94, 15, 31, 53, 9, 33, 25, 45, 7,
35, 40, 62, 49, 65, 60, 14, 88, 8, 50, 3, 27, 26, 13, 22, 52, 79, 83, 21, 98, 84, 93, 28, 44, 12, 37, 10
, 77, 47, 85, 100, 89, 16, 64, 71]
>>>
```

Figure 4: Output when Depth First Search was implemented on net3

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1a.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684
/Lab 9')
Reloaded modules: net3
The sequence of nodes in the order they are visited are:  [1, 267, 219, 340, 431, 210, 116, 87, 12, 319,
354, 212, 176, 208, 500, 296, 303, 172, 8, 338, 52, 38, 101, 476, 467, 46, 372, 97, 333, 236, 41, 152, 1
98, 197, 249, 436, 318, 125, 393, 40, 169, 496, 2, 3, 206, 484, 171, 317, 199, 83, 392, 133, 440, 53, 28
4, 409, 275, 135, 222, 107, 100, 413, 122, 5, 245, 304, 364, 286, 342, 293, 355, 30, 390, 389, 148, 307,
414, 58, 104, 20, 305, 381, 283, 426, 388, 126, 56, 270, 450, 108, 62, 168, 42, 326, 131, 88, 184, 347,
327, 433, 145, 451, 31, 164, 44, 240, 367, 29, 70, 415, 497, 273, 320, 322, 411, 428, 92, 45, 310, 383,
289, 47, 373, 366, 323, 309, 195, 149, 472, 139, 399, 15, 441, 299, 141, 134, 33, 96, 94, 438, 344, 181,
300, 128, 446, 196, 79, 21, 123, 384, 190, 271, 18, 290, 35, 481, 115, 298, 189, 348, 329, 118, 250, 321
, 266, 160, 395, 216, 356, 380, 442, 429, 443, 447, 105, 475, 482, 226, 295, 93, 361, 370, 69, 279, 288,
377, 445, 483, 153, 229, 202, 32, 98, 75, 369, 50, 396, 65, 106, 491, 453, 335, 460, 14, 345, 276, 43, 3
41, 81, 401, 166, 256, 191, 308, 404, 201, 313, 78, 90, 379, 182, 177, 36, 253, 287, 262, 499, 86, 432,
391, 465, 385, 274, 121, 34, 365, 218, 54, 138, 418, 255, 146, 352, 231, 306, 457, 421, 207, 371, 16, 6,
9, 203, 85, 142, 67, 157, 297, 448, 407, 194, 492, 234, 4, 251, 420, 403, 449, 269, 280, 63, 478, 339, 6
8, 487, 278, 281, 84, 49, 137, 419, 444, 180, 224, 351, 19, 37, 439, 257, 268, 360, 314, 336, 232, 209,
235, 498, 112, 186, 473, 332, 51, 156, 205, 254, 119, 324, 452, 17, 485, 353, 277, 57, 412, 479, 468, 20
0, 102, 258, 192, 316, 458, 386, 215, 204, 410, 159, 174, 22, 346, 178, 337, 26, 376, 397, 408, 405, 28,
263, 246, 488, 402, 242, 117, 282, 423, 163, 214, 24, 113, 489, 143, 161, 132, 103, 130, 334, 454, 325,
144, 179, 400, 60, 66, 221, 39, 424, 474, 77, 434, 239, 228, 185, 285, 435, 350, 82, 140, 187, 359, 312,
72, 150, 471, 437, 225, 265, 311, 470, 486, 175, 387, 27, 331, 466, 291, 59, 455, 427, 61, 490, 158, 95,
136, 301, 173, 237, 261, 493, 211, 7, 259, 151, 260, 248, 459, 193, 362, 375, 430, 55, 217, 165, 469, 16
2, 11, 292, 241, 230, 25, 294, 23, 315, 247, 238, 220, 147, 48, 477, 461, 154, 343, 417, 264, 120, 244,
167, 89, 91, 416, 155, 10, 114, 71, 213, 252, 374, 480, 170, 357, 243, 363, 233, 349, 76, 330, 463, 328,
302, 272, 456, 111, 422, 378, 382, 64, 73, 398, 223, 358, 464, 99, 124, 127, 129, 183, 110, 368, 109, 74
, 425, 462, 494, 80, 188, 406, 394, 227, 13, 495]
>>>
```

Figure 5: Output when Depth First Search was implemented on net4

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1a.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684
/Lab 9')
Reloaded modules: net4
The sequence of nodes in the order they are visited are:  [1, 298, 499, 255, 335, 281, 49, 488, 51, 101,
419, 343, 188, 185, 119, 485, 30, 274, 238, 301, 23, 53, 98, 132, 338, 496, 493, 413, 91, 201, 160, 154,
270, 250, 348, 177, 325, 308, 172, 454, 479, 268, 370, 22, 468, 397, 442, 359, 446, 149, 408, 355, 129,
365, 248, 475, 229, 92, 445, 447, 222, 241, 110, 378, 137, 443, 102, 239, 240, 203, 117, 394, 183, 275,
212, 297, 487, 293, 211, 24, 70, 39, 19, 438, 374, 66, 116, 489, 164, 425, 141, 369, 18, 497, 186, 50, 4
24, 252, 271, 15, 269, 26, 300, 344, 441, 322, 392, 218, 230, 180, 120, 417, 389, 307, 20, 60, 380, 398,
166, 409, 354, 38, 175, 469, 206, 21, 184, 56, 440, 112, 34, 402, 412, 470, 99, 319, 377, 455, 368, 318,
260, 13, 279, 459, 84, 363, 79, 463, 400, 135, 367, 264, 253, 216, 331, 494, 161, 312, 481, 311, 390, 25
9, 257, 258, 432, 162, 220, 498, 321, 6, 288, 315, 291, 176, 473, 353, 65, 434, 361, 342, 474, 140, 482,
362, 285, 159, 181, 299, 391, 347, 88, 225, 472, 421, 332, 71, 352, 42, 32, 208, 199, 73, 33, 75, 492, 4
57, 193, 227, 439, 94, 309, 80, 179, 46, 62, 7, 456, 37, 324, 97, 100, 458, 280, 232, 327, 10, 330, 95,
371, 204, 126, 224, 420, 105, 178, 104, 350, 437, 243, 121, 364, 17, 272, 379, 366, 81, 235, 226, 414, 4
83, 111, 171, 47, 303, 376, 108, 404, 375, 139, 205, 323, 147, 244, 406, 5, 334, 316, 78, 86, 130, 134,
381, 67, 48, 345, 476, 426, 57, 143, 106, 85, 170, 41, 153, 478, 411, 329, 295, 2, 157, 29, 168, 4, 336,
430, 125, 128, 103, 387, 491, 276, 405, 477, 8, 109, 471, 192, 422, 90, 461, 245, 306, 407, 444, 76, 209
, 234, 167, 358, 385, 423, 118, 31, 261, 198, 410, 163, 433, 58, 16, 219, 74, 236, 292, 451, 333, 436, 4
84, 138, 490, 68, 265, 195, 114, 14, 59, 246, 207, 215, 242, 349, 500, 256, 403, 156, 107, 83, 302, 339,
284, 462, 93, 69, 435, 480, 287, 263, 393, 151, 283, 373, 495, 356, 131, 54, 158, 346, 133, 197, 189, 45
, 63, 182, 122, 64, 464, 294, 337, 448, 415, 155, 55, 416, 200, 145, 146, 384, 233, 25, 289, 466, 399, 3
10, 213, 52, 202, 351, 210, 341, 191, 465, 217, 286, 165, 196, 148, 72, 77, 282, 11, 28, 395, 266, 431,
429, 194, 328, 278, 36, 142, 27, 388, 231, 35, 418, 486, 61, 449, 360, 357, 396, 383, 267, 87, 152, 450,
223, 305, 296, 9, 12, 277, 96, 452, 136, 40, 187, 221, 254, 326, 428, 251, 43, 150, 304, 124, 313, 290,
453, 401, 82, 173, 174, 372, 127, 467, 427, 262, 340, 273, 314, 169, 44, 237, 320, 386, 3, 247, 113, 382
, 115, 214, 228, 144, 317, 460, 123, 89, 190, 249]
>>>
```

Figure 6: Output when Depth First Search was implemented on net5

```
The sequence of nodes in the order they are visited are:  [1, 89, 626, 680, 432, 792, 610, 583, 633, 400
, 622, 573, 396, 6, 259, 265, 537, 380, 540, 190, 555, 106, 595, 366, 41, 8, 340, 644, 104, 751, 552, 31
, 149, 274, 209, 497, 598, 594, 156, 100, 472, 782, 781, 449, 344, 757, 697, 232, 565, 253, 733, 121, 27
3, 705, 482, 279, 158, 464, 7, 756, 738, 301, 615, 111, 82, 239, 47, 567, 663, 356, 155, 323, 480, 529,
307, 53, 658, 172, 698, 52, 458, 238, 397, 768, 451, 422, 163, 437, 168, 233, 707, 590, 252, 174, 547, 3
06, 57, 702, 486, 194, 122, 614, 370, 167, 447, 421, 78, 548, 312, 210, 519, 302, 562, 283, 763, 136, 24
9, 744, 242, 188, 219, 551, 367, 3, 683, 161, 750, 487, 686, 214, 40, 216, 56, 243, 550, 38, 494, 79, 28
7, 313, 300, 553, 299, 736, 666, 305, 645, 762, 86, 48, 341, 517, 729, 112, 250, 346, 295, 288, 745, 641
, 409, 378, 720, 191, 14, 126, 647, 339, 521, 728, 591, 114, 244, 526, 25, 196, 399, 365, 165, 539, 471,
50, 353, 71, 462, 429, 83, 784, 514, 541, 585, 640, 741, 248, 679, 142, 355, 203, 627, 213, 140, 412, 45
6, 522, 469, 77, 391, 725, 388, 402, 688, 229, 709, 629, 251, 236, 408, 60, 226, 202, 342, 55, 193, 722,
74, 352, 685, 419, 269, 230, 625, 676, 91, 291, 485, 600, 542, 710, 438, 515, 577, 267, 201, 554, 632, 6
52, 208, 215, 696, 246, 221, 568, 5, 350, 798, 153, 674, 746, 755, 530, 42, 49, 128, 212, 571, 286, 360,
719, 454, 596, 677, 560, 617, 428, 481, 669, 182, 115, 16, 463, 777, 799, 490, 491, 359, 656, 260, 315,
262, 138, 120, 618, 374, 17, 205, 473, 351, 662, 692, 589, 436, 574, 383, 580, 257, 198, 660, 681, 488,
783, 43, 770, 189, 379, 389, 61, 254, 636, 116, 272, 637, 416, 465, 780, 678, 390, 461, 385, 433, 499, 4
60, 34, 535, 477, 439, 309, 92, 753, 619, 59, 735, 511, 597, 440, 584, 420, 176, 502, 503, 630, 264, 778
, 2, 713, 643, 256, 425, 124, 175, 135, 332, 195, 766, 661, 335, 701, 413, 67, 684, 406, 723, 435, 18, 3
21, 392, 139, 523, 544, 197, 73, 635, 160, 452, 648, 308, 659, 373, 706, 754, 703, 171, 475, 579, 531, 1
37, 649, 20, 711, 277, 148, 566, 330, 84, 403, 775, 445, 726, 141, 479, 44, 169, 29, 268, 125, 556, 278,
734, 655, 362, 266, 536, 90, 51, 211, 394, 96, 110, 129, 786, 558, 186, 700, 10, 282, 93, 377, 581, 331,
650, 747, 372, 749, 35, 143, 234, 776, 527, 690, 689, 670, 4, 363, 62, 298, 569, 601, 154, 606, 173, 699
, 101, 303, 765, 145, 384, 170, 450, 13, 337, 157, 496, 431, 773, 800, 133, 470, 605, 742, 448, 285, 476
, 760, 23, 538, 715, 759, 75, 371, 739, 459, 727, 631, 159, 602, 624, 500, 509, 280, 516, 348, 524, 545,
667, 123, 687, 46, 217, 444, 732, 513, 410, 108, 376, 316, 708, 493, 790, 468, 534, 563, 375, 411, 668,
673, 675, 518, 769, 730, 691, 712, 672, 329, 576, 620, 72, 405, 247, 147, 88, 358, 794, 204, 506, 785, 7
71, 724, 151, 386, 758, 623, 349, 99, 787, 761, 593, 354, 107, 642, 797, 12, 752, 609, 24, 628, 582, 718
, 357, 651, 183, 343, 793, 296, 30, 796, 501, 64, 220, 222, 611, 543, 150, 418, 327, 484, 225, 414, 387,
338, 293, 599, 748, 364, 430, 237, 200, 731, 206, 772, 612, 45, 180, 76, 334, 369, 789, 192, 572, 131, 5
92, 474, 297, 290, 11, 270, 21, 404, 87, 682, 466, 130, 578, 740, 224, 743, 561, 9, 382, 455, 368, 152,
704, 347, 764, 336, 117, 381, 613, 653, 693, 164, 113, 85, 144, 326, 310, 646, 767, 58, 721, 441, 492, 7
37, 621, 80, 119, 557, 694, 665, 178, 318, 407, 235, 146, 325, 559, 549, 66, 478, 184, 317, 324, 508, 42
6, 32, 467, 103, 453, 207, 564, 774, 284, 607, 457, 94, 162, 520, 292, 532, 788, 795, 525, 533, 345, 505
, 102, 109, 179, 311, 322, 638, 304, 361, 28, 634, 604, 512, 65, 716, 657, 528, 791, 27, 199, 95, 415, 3
6, 575, 271, 177, 664, 19, 240, 245, 393, 258, 185, 132, 276, 228, 275, 333, 654, 33, 695, 639, 616, 446
, 570, 717, 424, 588, 320, 498, 489, 423, 187, 507, 314, 401, 241, 417, 714, 227, 105, 127, 181, 395, 44
2, 231, 289, 15, 118, 263, 54, 779, 587, 70, 586, 255, 603, 294, 218, 510, 22, 608, 81, 134, 69, 443, 26
, 39, 37, 427, 98, 223, 97, 63, 483, 398, 671, 328, 166, 319, 504, 546, 68, 281, 495, 261, 434]
>>>
```

Figure 7: Output when Depth First Search was implemented on net6

Subpart b:

File: ex1c.py

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net0
The sequence of nodes in the order they are visited are:  [1, 5, 3, 2, 4, 2, 4, 3, 2]
Cycle Present!
>>>
```

Figure 8: Output when Depth First Search was implemented on net0 for checking whether graph has a cycle or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net0
The sequence of nodes in the order they are visited are:  [1, 2, 6, 9, 7, 5, 3, 10, 4, 8, 4, 10, 4,
10, 7, 5, 3, 8, 7, 9, 10]
Cycle Present!
>>>
```

Figure 9: Output when Depth First Search was implemented on net1 for checking whether graph has a cycle or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net1
The sequence of nodes in the order they are visited are:  [1, 29, 39, 13, 12, 49, 48, 3, 50, 7, 40,
11, 36, 15, 43, 26, 6, 31, 21, 2, 22, 37, 20, 16, 27, 9, 47, 44, 19, 34, 38, 32, 5, 10, 42, 30, 14,
17, 46, 33, 33, 41, 35, 33, 4, 23, 8, 17, 46, 18, 4, 14, 9, 41, 27, 33, 33, 8, 16, 45, 28, 45, 17,
8, 14, 10, 25, 44, 37, 10, 24, 21, 20, 9, 14, 5, 24, 15, 33, 32, 25, 26, 34, 27, 37, 43, 38, 7, 30,
4, 22, 31, 7, 22, 31, 18, 11, 31, 41, 45, 46]
Cycle Present!
>>>
```

Figure 10: Output when Depth First Search was implemented on net2 for checking whether graph has a cycle or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net3
Cycle Present!
>>>
```

Figure 11: Output when Depth First Search was implemented on net3 for checking whether graph has a cycle or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net4
Cycle Present!
>>>
```

Figure 12: Output when Depth First Search was implemented on net4 for checking whether graph has a cycle or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net5
Cycle Present!
```

Figure 13: Output when Depth First Search was implemented on net5 for checking whether graph has a cycle or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1c.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net6
Cycle Present!
>>>
```

Figure 14: Output when Depth First Search was implemented on net6 for checking whether graph has a cycle or not

Subpart d:

File: ex1d.py

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 6
84/Lab 9')
The sequence of nodes in the order they are visited are:  [1, 5, 3, 2, 4]
Graph is connected!
>>>
```

Figure 15: Output when Depth First Search was implemented on net0 for checking whether graph is connected or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net1
The sequence of nodes in the order they are visited are:  [1, 2, 6, 9, 7, 5, 3, 10, 4, 8]
Graph is connected!
>>>
```

Figure 16: Output when Depth First Search was implemented on net1 for checking whether graph is connected or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net2
The sequence of nodes in the order they are visited are:  [1, 29, 39, 13, 12, 49, 48, 3, 50, 7, 40,
11, 36, 15, 43, 26, 6, 31, 21, 2, 22, 37, 20, 16, 27, 9, 47, 44, 19, 34, 38, 32, 5, 10, 42, 30, 14,
17, 46, 33, 41, 35, 4, 23, 8, 18, 45, 28, 25, 24]
Graph is connected!
>>>
```

Figure 17: Output when Depth First Search was implemented on net2 for checking whether graph is connected or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net3
The sequence of nodes in the order they are visited are:  [1, 75, 96, 34, 99, 74, 67, 92, 24, 11, 6
1, 81, 41, 69, 59, 72, 23, 4, 86, 87, 2, 78, 73, 82, 90, 91, 76, 80, 36, 19, 42, 38, 46, 32, 30, 56
, 58, 68, 6, 17, 51, 5, 70, 39, 43, 66, 54, 20, 48, 29, 95, 97, 57, 63, 55, 18, 94, 15, 31, 53, 9,
33, 25, 45, 7, 35, 40, 62, 49, 65, 60, 14, 88, 8, 50, 3, 27, 26, 13, 22, 52, 79, 83, 21, 98, 84, 93
, 28, 44, 12, 37, 10, 77, 47, 85, 100, 89, 16, 64, 71]
Graph is connected!
>>>
```

Figure 18: Output when Depth First Search was implemented on net3 for checking whether graph is connected or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net4
Graph is connected!
>>>
```

Figure 19: Output when Depth First Search was implemented on net4 for checking whether graph is connected or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net5
Graph is connected!
>>>
```

Figure 20: Output when Depth First Search was implemented on net5 for checking whether graph is connected or not

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex1d.py', wdir='C:/Users/Aak/Desktop/Sem IV/I
E 684/Lab 9')
Reloaded modules: net6
Graph is connected!
>>>
```

Figure 21: Output when Depth First Search was implemented on net6 for checking whether graph is connected or not

Question 2:

Subpart a:

Working of Prim's Algorithm:
In this algorithm we make a matrix M of edge lengths where M_{ij} denotes the edge length between nodes i and j. Now, we start by taking the edge with minimum edge weight in the first row, and add it to our MST and marking that particular edge as 'taken'. In the second iteration, we take the edge corresponding to the minimum 'non-taken' edge weight in the first TWO rows (such that there is no cycle in the MST after the edge gets added) and add it to our MST along with marking the edge as 'taken'.

Proceeding in this way, at the k^{th} iteration, we take the edge corresponding to the minimum 'non-taken' edge weight in the first K rows (such that there is no cycle in the MST after the edge gets added) and add it to our MST along with marking the edge as 'taken'.

After n-1 iterations we would have our MST.

Working of Kruskal's Algorithm:
In this algorithm we sort each edge in increasing order of their edge-weights. We run the algorithm $|E|$ times. In iteration i we check whether adding the i^{th} smallest edge into our MST creates a cycle or not. If not, we add that particular edge in our MST. Proceeding this way, we would get out MST.

Subpart b:

File: ex2b.py

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex2b.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9')
Reloaded modules: net0
The edge between nodes 1 and 2 with weight 14 is in the MST.
The edge between nodes 1 and 3 with weight 15 is in the MST.
The edge between nodes 3 and 5 with weight 11 is in the MST.
The edge between nodes 1 and 5 with weight 17 will not be in the MST or else there will be a cycle.
The edge between nodes 4 and 1 with weight 19 is in the MST.
The total edge weight in the MST is: 59
>>>
```

Figure 22: Output when Prim's Algorithm for net0

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex2b.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9')
Reloaded modules: net1
The edge between nodes 1 and 7 with weight 12 is in the MST.
The edge between nodes 1 and 9 with weight 13 is in the MST.
The edge between nodes 3 and 4 with weight 12 is in the MST.
The edge between nodes 3 and 5 with weight 13 is in the MST.
The edge between nodes 2 and 1 with weight 15 is in the MST.
The edge between nodes 5 and 7 with weight 16 is in the MST.
The edge between nodes 2 and 10 with weight 20 is in the MST.
The edge between nodes 3 and 1 with weight 21 will not be in the MST or else there will be a cycle.
The edge between nodes 8 and 4 with weight 22 is in the MST.
The edge between nodes 9 and 7 with weight 24 will not be in the MST or else there will be a cycle.
The edge between nodes 5 and 2 with weight 29 will not be in the MST or else there will be a cycle.
The edge between nodes 8 and 1 with weight 31 will not be in the MST or else there will be a cycle.
The edge between nodes 9 and 6 with weight 32 is in the MST.
The total edge weight in the MST is: 155
>>>
```

Figure 23: Output when Prim's Algorithm for net1

```
The edge between nodes 17 and 32 with weight 12 is in the MST.
The edge between nodes 18 and 39 with weight 11 is in the MST.
The edge between nodes 17 and 46 with weight 14 is in the MST.
The edge between nodes 13 and 12 with weight 17 is in the MST.
The edge between nodes 16 and 9 with weight 19 is in the MST.
The edge between nodes 10 and 36 with weight 20 is in the MST.
The edge between nodes 23 and 32 with weight 10 is in the MST.
The edge between nodes 24 and 36 with weight 19 is in the MST.
The edge between nodes 24 and 40 with weight 21 is in the MST.
The edge between nodes 26 and 43 with weight 14 is in the MST.
The edge between nodes 26 and 50 with weight 16 is in the MST.
The edge between nodes 28 and 45 with weight 20 is in the MST.
The edge between nodes 27 and 9 with weight 21 is in the MST.
The edge between nodes 30 and 42 with weight 13 is in the MST.
The edge between nodes 31 and 45 with weight 10 is in the MST.
The edge between nodes 31 and 39 with weight 13 is in the MST.
The edge between nodes 33 and 37 with weight 23 is in the MST.
The edge between nodes 34 and 38 with weight 17 is in the MST.
The edge between nodes 34 and 48 with weight 23 is in the MST.
The edge between nodes 22 and 13 with weight 24 is in the MST.
The edge between nodes 37 and 20 with weight 24 is in the MST.
The edge between nodes 36 and 21 with weight 25 is in the MST.
The edge between nodes 19 and 44 with weight 25 is in the MST.
The edge between nodes 33 and 46 with weight 26 is in the MST.
The edge between nodes 41 and 20 with weight 26 is in the MST.
The edge between nodes 20 and 16 with weight 26 will not be in the MST or else there will be a cycle.
The edge between nodes 15 and 43 with weight 26 is in the MST.
The edge between nodes 39 and 22 with weight 27 is in the MST.
The edge between nodes 44 and 47 with weight 14 is in the MST.
The edge between nodes 36 and 20 with weight 27 will not be in the MST or else there will be a cycle.
The edge between nodes 17 and 14 with weight 27 will not be in the MST or else there will be a cycle.
The edge between nodes 13 and 7 with weight 29 is in the MST.
The edge between nodes 39 and 11 with weight 31 is in the MST.
The edge between nodes 34 and 19 with weight 31 is in the MST.
The edge between nodes 48 and 49 with weight 30 is in the MST.
The edge between nodes 40 and 7 with weight 32 is in the MST.
The total edge weight in the MST is: 963
>>>
```

Figure 24: Output when Prim's Algorithm for net2

```
The total edge weight in the MST is: 2021
>>>
```

Figure 25: Output when Prim's Algorithm for net3

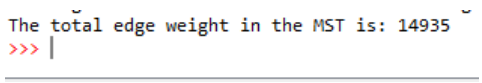

Figure 26: Output when Prim's Algorithm for net4

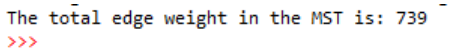

Figure 27: Output when Prim's Algorithm for net5

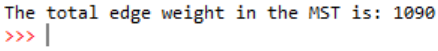

Figure 28: Output when Prim's Algorithm for net6

Subpart c:

File: ex2c.py

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex2c.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9')
Reloaded modules: net0
The edge between nodes 3 and 5 with weight 11 is in the MST.
The edge between nodes 1 and 2 with weight 14 is in the MST.
The edge between nodes 1 and 3 with weight 15 is in the MST.
The edge between nodes 1 and 5 with weight 17 will not be in the MST or else there will be a cycle.
The edge between nodes 1 and 4 with weight 19 is in the MST.
The edge between nodes 2 and 5 with weight 21 will not be in the MST or else there will be a cycle.
The edge between nodes 2 and 3 with weight 23 will not be in the MST or else there will be a cycle.
The edge between nodes 2 and 4 with weight 26 will not be in the MST or else there will be a cycle.
The total edge weight in the MST is: 59
>>>
```

Figure 29: Output when Kruskal's Algorithm for net0

```
>>> runfile('C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9/ex2c.py', wdir='C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 9')
Reloaded modules: net1
The edge between nodes 3 and 4 with weight 12 is in the MST.
The edge between nodes 1 and 7 with weight 12 is in the MST.
The edge between nodes 3 and 5 with weight 13 is in the MST.
The edge between nodes 1 and 9 with weight 13 is in the MST.
The edge between nodes 1 and 2 with weight 15 is in the MST.
The edge between nodes 5 and 7 with weight 16 is in the MST.
The edge between nodes 2 and 10 with weight 20 is in the MST.
The edge between nodes 1 and 3 with weight 21 will not be in the MST or else there will be a cycle.
The edge between nodes 4 and 8 with weight 22 is in the MST.
The edge between nodes 7 and 9 with weight 24 will not be in the MST or else there will be a cycle.
The edge between nodes 2 and 5 with weight 29 will not be in the MST or else there will be a cycle.
The edge between nodes 1 and 8 with weight 31 will not be in the MST or else there will be a cycle.
The edge between nodes 4 and 6 with weight 32 is in the MST.
The edge between nodes 6 and 9 with weight 32 will not be in the MST or else there will be a cycle.
The edge between nodes 1 and 10 with weight 33 will not be in the MST or else there will be a cycle.
The edge between nodes 2 and 7 with weight 34 will not be in the MST or else there will be a cycle.
The edge between nodes 2 and 6 with weight 34 will not be in the MST or else there will be a cycle.
The edge between nodes 9 and 10 with weight 36 will not be in the MST or else there will be a cycle.
The edge between nodes 3 and 10 with weight 37 will not be in the MST or else there will be a cycle.
The edge between nodes 4 and 10 with weight 38 will not be in the MST or else there will be a cycle.
The total edge weight in the MST is: 155
>>>
```

Figure 30: Output when Kruskal's Algorithm for net1

```
The edge between nodes 32 and 38 with weight 37 is in the MST.
The edge between nodes 7 and 15 with weight 37 is in the MST.
The edge between nodes 17 and 41 with weight 38 is in the MST.
The edge between nodes 2 and 22 with weight 38 is in the MST.
The edge between nodes 30 and 33 with weight 39 is in the MST.
The edge between nodes 6 and 31 with weight 39 is in the MST.
The edge between nodes 12 and 43 with weight 39 is in the MST.
The edge between nodes 12 and 49 with weight 39 is in the MST.
The edge between nodes 29 and 39 with weight 39 is in the MST.
The edge between nodes 27 and 48 with weight 40 is in the MST.
The edge between nodes 17 and 33 with weight 41 is in the MST.
The edge between nodes 10 and 42 with weight 41 is in the MST.
The edge between nodes 8 and 22 with weight 42 is in the MST.
The edge between nodes 7 and 12 with weight 42 is in the MST.
The edge between nodes 2 and 16 with weight 43 is in the MST.
The edge between nodes 17 and 31 with weight 43 is in the MST.
The edge between nodes 35 and 41 with weight 44 is in the MST.
The edge between nodes 36 and 37 with weight 44 is in the MST.
The edge between nodes 9 and 18 with weight 44 is in the MST.
The edge between nodes 15 and 25 with weight 44 is in the MST.
The edge between nodes 1 and 29 with weight 44 will not be in the MST or else there will be a cycle.
The edge between nodes 8 and 26 with weight 45 is in the MST.
The edge between nodes 12 and 30 with weight 46 is in the MST.
The edge between nodes 7 and 32 with weight 46 is in the MST.
The edge between nodes 2 and 45 with weight 47 is in the MST.
The edge between nodes 14 and 43 with weight 47 is in the MST.
The edge between nodes 1 and 45 with weight 47 will not be in the MST or else there will be a cycle.
The edge between nodes 7 and 33 with weight 47 is in the MST.
The edge between nodes 21 and 31 with weight 48 is in the MST.
The edge between nodes 3 and 50 with weight 48 is in the MST.
The edge between nodes 11 and 40 with weight 48 is in the MST.
The edge between nodes 5 and 40 with weight 49 is in the MST.
The edge between nodes 29 and 31 with weight 49 is in the MST.
The edge between nodes 22 and 37 with weight 49 is in the MST.
The edge between nodes 13 and 31 with weight 50 is in the MST.
The edge between nodes 20 and 27 with weight 50 is in the MST.
The total edge weight in the MST is: 2871
>>>
```

Figure 31: Output when Kruskal's Algorithm for net2

```
The total edge weight in the MST is: 2021
>>>
```

Figure 32: Output when Kruskal's Algorithm for net3

```
The total edge weight in the MST is: 14923
>>>
```

Figure 33: Output when Kruskal's Algorithm for net4

```
The total edge weight in the MST is: 739
>>>
```

Figure 34: Output when Kruskal's Algorithm for net5

```
The total edge weight in the MST is: 1188
>>>
```

Figure 35: Output when Kruskal's Algorithm for net6