



Tecnológico de Monterrey

Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

Homework 10: BnB - Welsh Powell

Group: 607
Team #3

Luis Salomón Flores Ugalde

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

FileEditSelectionViewGoRunTerminalHelp←→Q 2.Advanced Algorithms

HW10_BnBWP.py ×

Homework10BnB_WP > HW10_BnBWP.py > ...
1 # Analysis and Design of Advanced Algorithms
2 # Group #607
3 # Team 3
4 # Luis Salomón Flores Ugalde
5
6 # Santiago Quintana Moreno A01571222
7 # Miguel Ángel Álvarez Hermida A01722925
8
9 # ----- Homework 10 B&B and Welsh-Powell -----
10 from collections import deque
11
12 # ---- Build Graph ----
13 v = ["A", "B", "C", "D", "E"]
14 edges = {
15 ("A", "B"),
16 ("A", "C"),
17 ("A", "D"),
18 ("A", "E"),
19 ("B", "E"),
20 ("C", "D"),
21 ("D", "E"),
22 }
23
24 def build_adj(vertices, edge_set):
25 adj = {v: set() for v in vertices}
26 for u, v in edge_set:
27 adj[u].add(v)
28 adj[v].add(u)
29 return adj
30
31 ADJ = build_adj(v, edges)
32
33 # ---- Welsh-Powell Coloring (greedy by degree) ----
34 def welsch_powell_coloring(adj):
35 order = sorted(adj, key=lambda v: len(adj[v]), reverse=True)
36 colors = {}
37 color_id = 0
38 for v in order:
39 if v in colors:
40 continue
41 colors[v] = color_id
42 for u in order:
43 if u not in colors and (v, u) not in edges:
44 colors[u] = color_id
45 color_id += 1
46 return colors

powershell ×

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework10BnB_WP/Hw10_BnBWP.py"
Vertices: ['A', 'B', 'C', 'D', 'E']
Edges: [('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'E'), ('C', 'D'), ('D', 'E')]

Welsh-Powell coloring (#colors = 3)
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

Branch & Bound minimum coloring (#colors = 3)
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

All-pairs shortest path distances (hops)
A B C D E
A 0 1 1 1 1
B 1 0 2 2 1
C 1 2 0 1 2
D 1 2 1 0 1
E 1 1 2 1 0

Example shortest paths:
A->B: A-B (hops=1)
A->C: A-C (hops=1)
A->D: A-D (hops=1)
A->E: A-E (hops=1)

B->A: B-A (hops=1)
B->C: B-A-C (hops=2)
B->D: B-E-D (hops=2)
B->E: B-E (hops=1)

C->A: C-A (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->E: C-D-E (hops=2)

main ↺ 🔍 0 △ 0

FileEditSelectionViewGoRunTerminalHelp←→Q 2.Advanced Algorithms

HW10_BnBWP.py ×

Homework10BnB_WP > HW10_BnBWP.py > ...

```
34 def welsh_powell_coloring(adj):
40     continue
41     colors[v] = color_id
42     for u in order:
43         if u in colors:
44             continue
45         if all(u not in adj[w] for w, c in colors.items() if c == color_id):
46             colors[u] = color_id
47         color_id += 1
48     return colors, color_id
49
50 # ---- Branch & Bound (minimum coloring) ----
51 def _greedy_upper_bound(adj):
52     order = sorted(adj, key=lambda v: len(adj[v]), reverse=True)
53     color = {}
54     for v in order:
55         taken = {color[u] for u in adj[v] if u in color}
56         c = 0
57         while c in taken:
58             c += 1
59         color[v] = c
60     return color, max(color.values()) + 1
61
62 def branch_and_bound_coloring(adj):
63     order = sorted(adj, key=lambda v: len(adj[v]), reverse=True)
64     best_coloring, best_k = _greedy_upper_bound(adj)
65
66     color = {v: None for v in adj}
67     used_colors = 0
68
69     def assign(i):
70         nonlocal best_coloring, best_k, used_colors
71         if i == len(order):
72             k = used_colors
73             if k < best_k:
74                 best_k = k
75                 best_coloring = color.copy()
76             return
77         v = order[i]
78         for c in range(used_colors):
79             if all(color[u] != c for u in adj[v] if color[u] is not None):
80                 color[v] = c
```

powershell ×

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework10BnB_WP/HW10_BnBWP.py"

Vertices: ['A', 'B', 'C', 'D', 'E']

Edges: [('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'E'), ('C', 'D'), ('D', 'E')]

Welsh-Powell coloring (#colors = 3)

A: color 0

B: color 1

C: color 2

D: color 1

E: color 2

Branch & Bound minimum coloring (#colors = 3)

A: color 0

B: color 1

C: color 2

D: color 1

E: color 2

All-pairs shortest path distances (hops)

	A	B	C	D	E
A	0	1	1	1	1
B	1	0	2	2	1
C	1	2	0	1	2
D	1	2	1	0	1
E	1	1	2	1	0

Example shortest paths:

A->B: A-B (hops=1)

A->C: A-C (hops=1)

A->D: A-D (hops=1)

A->E: A-E (hops=1)

B->A: B-A (hops=1)

B->C: B-A-C (hops=2)

B->D: B-E-D (hops=2)

B->E: B-E (hops=1)

C->A: C-A (hops=1)

C->B: C-A-B (hops=2)

C->D: C-D (hops=1)

C->E: C-A-E (hops=2)

D->A: D-A (hops=1)

D->B: D-A-B (hops=2)

D->C: D-A-C (hops=2)

D->E: D-E (hops=1)

E->A: E-A (hops=1)

E->B: E-B (hops=1)

E->C: E-A-C (hops=2)

E->D: E-D (hops=1)

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

HW10_BnBWP.py ×

Homework10BnB_WP > HW10_BnBWP.py > ...
62 def branch_and_bound_coloring(adj):
63 def assign(i):
64 if color[u] is not None and adj[u][i] is not None:
65 color[v] = c
66 assign(i + 1)
67 color[v] = None
68 if used_colors + 1 < best_k:
69 c = used_colors
70 color[v] = c
71 used_colors += 1
72 assign(i + 1)
73 used_colors -= 1
74 color[v] = None
75 assign(0)
76 remap, nextc = {}, 0
77 normalized = {}
78 for v in order:
79 c = best_coloring[v]
80 if c not in remap:
81 remap[c] = nextc
82 nextc += 1
83 normalized[v] = remap[c]
84 return normalized, best_k
85
86 # ---- All-Pairs Shortest Paths (unweighted, BFS per source) ----
87 def bfs_distances(adj, src):
88 dist = {v: float("inf") for v in adj}
89 parent = {v: None for v in adj}
90 dist[src] = 0
91 q = deque([src])
92 while q:
93 v = q.popleft()
94 for u in adj[v]:
95 if dist[u] == float("inf"):
96 dist[u] = dist[v] + 1
97 parent[u] = v
98 q.append(u)
99 return dist, parent
100
101 def all_pairs_shortest_paths(adj, vertices):
102 all_dist = {}
103 all_paths = {}
104 for v in vertices:
105 dist, parent = bfs_distances(adj, v)
106 all_dist[v] = dist
107 all_paths[v] = parent
108 return all_dist, all_paths

...

powershell ×

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework10BnB_WP/HW10_BnBWP.py"
Vertices: ['A', 'B', 'C', 'D', 'E']
Edges: [('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'E'), ('C', 'D'), ('D', 'E')]

Welsh-Powell coloring (#colors = 3)
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

Branch & Bound minimum coloring (#colors = 3)
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

All-pairs shortest path distances (hops)
A B C D E
A 0 1 1 1 1
B 1 0 2 2 1
C 1 2 0 1 2
D 1 2 1 0 1
E 1 1 2 1 0

Example shortest paths:
A->B: A-B (hops=1)
A->C: A-C (hops=1)
A->D: A-D (hops=1)
A->E: A-E (hops=1)

B->A: B-A (hops=1)
B->C: B-A-C (hops=2)
B->D: B-E-D (hops=2)
B->E: B-E (hops=1)

C->A: C-A (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->E: C-A-E (hops=2)
C->D: C-D (hops=1)
C->E: C-D-E (hops=2)

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

HW10_BnBWP.py ×

Homework10BnB_WP > HW10_BnBWP.py > ...

```
117 def all_pairs_shortest_paths(adj, vertices):
118     all_dists = {}
119     all_paths = {}
120     for s in vertices:
121         dist, parent = bfs_distances(adj, s)
122         all_dists[s] = dist
123         paths_s = {}
124         for t in vertices:
125             if dist[t] == float("inf"):
126                 paths_s[t] = None
127             else:
128                 cur, path = t, []
129                 while cur is not None:
130                     path.append(cur)
131                     if cur == s:
132                         break
133                     cur = parent[cur]
134                 paths_s[t] = list(reversed(path))
135         all_paths[s] = paths_s
136     return all_dists, all_paths
137
138 # ---- Pretty printing helpers ----
139 def print_coloring(title, coloring, k):
140     print(f"{title} (#colors = {k})")
141     for v in sorted(coloring):
142         print(f"    {v}: color {coloring[v]}")
143     print()
144
145 def print_distance_matrix(vertices, all_dists):
146     header = "    " + " ".join(f"{v:>3}" for v in vertices)
147     print("All-pairs shortest path distances (hops)")
148     print(header)
149     for r in vertices:
150         row = [f"{all_dists[c][r]:>3}" for c in vertices]
151         print(f"{r:>3} " + " ".join(row))
152     print()
153
154 def print_example_paths(vertices, paths):
155     print("Example shortest paths:")
156     for s in vertices:
157         for t in vertices:
158             if s == t:
```

powershell ×

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework10BnB_WP/HW10_BnBWP.py"

Vertices: ['A', 'B', 'C', 'D', 'E']
Edges: [('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'E'), ('C', 'D'), ('D', 'E')]

Welsh-Powell coloring (#colors = 3)
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

Branch & Bound minimum coloring (#colors = 3)
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

All-pairs shortest path distances (hops)

	A	B	C	D	E
A	0	1	1	1	1
B	1	0	2	2	1
C	1	2	0	1	2
D	1	2	1	0	1
E	1	1	2	1	0

Example shortest paths:
A->B: A-B (hops=1)
A->C: A-C (hops=1)
A->D: A-D (hops=1)
A->E: A-E (hops=1)

B->A: B-A (hops=1)
B->C: B-A-C (hops=2)
B->D: B-E-D (hops=2)
B->E: B-E (hops=1)

C->A: C-A (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->E: C-A-E (hops=2)

D->A: D-A (hops=1)
D->B: D-A-B (hops=2)
D->C: D-C (hops=1)
D->E: D-E (hops=1)

E->A: E-A (hops=1)
E->B: E-B (hops=1)
E->C: E-A-C (hops=2)
E->D: E-D (hops=1)

main



HW10_BnBWP.py X

Homework10BnB_WP > HW10_BnBWP.py > ...

```
154 def print_example_paths(vertices, paths):
155     for s in vertices:
156         for t in vertices:
157             if s == t:
158                 continue
159             p = paths[s][t]
160             if p is None:
161                 print(f" {s}->{t}: no path")
162             else:
163                 print(f" {s}->{t}: {'-'.join(p)} (hops={len(p)-1})")
164             print()
165
166
167 if __name__ == "__main__":
168     print("Vertices:", v)
169     print("Edges:", sorted(tuple(sorted(e)) for e in edges))
170     print()
171
172     wp_colors, wp_k = welsh_powell_coloring(ADJ)
173     bb_colors, bb_k = branch_and_bound_coloring(ADJ)
174     all_dists, all_paths = all_pairs_shortest_paths(ADJ, v)
175
176     print_coloring("Welsh-Powell coloring", wp_colors, wp_k)
177     print_coloring("Branch & Bound minimum coloring", bb_colors, bb_k)
178     print_distance_matrix(v, all_dists)
179     print_example_paths(v, all_paths)
180
```

powershell X

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\
Microsoft\WindowsApps\python3.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algori
thms/Homework10BnB_WP/HW10_BnBWP.py"
```

Vertices: ['A', 'B', 'C', 'D', 'E']

Edges: [('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'E'), ('C', 'D'), ('D', 'E')]

Welsh-Powell coloring (#colors = 3)

```
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2
```

Branch & Bound minimum coloring (#colors = 3)

```
A: color 0
B: color 1
C: color 2
D: color 1
E: color 2
```

All-pairs shortest path distances (hops)

	A	B	C	D	E
A	0	1	1	1	1
B	1	0	2	2	1
C	1	2	0	1	2
D	1	2	1	0	1
E	1	1	2	1	0

Example shortest paths:

```
A->B: A-B (hops=1)
A->C: A-C (hops=1)
A->D: A-D (hops=1)
A->E: A-E (hops=1)
```

```
B->A: B-A (hops=1)
B->C: B-A-C (hops=2)
B->D: B-E-D (hops=2)
B->E: B-E (hops=1)
```

```
C->A: C-A (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->E: C-D-E (hops=2)
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
\Microsoft\WindowsApps\python3.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algori
thms/Homework10BnB_WP/HW10_BnBWP.py"
Vertices: ['A', 'B', 'C', 'D', 'E']
Edges: [('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'E'), ('C', 'D'), ('D', 'E')]
```

Welsh-Powell coloring (#colors = 3)

A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

Branch & Bound minimum coloring (#colors = 3)

A: color 0
B: color 1
C: color 2
D: color 1
E: color 2

All-pairs shortest path distances (hops)

	A	B	C	D	E
A	0	1	1	1	1
B	1	0	2	2	1
C	1	2	0	1	2
D	1	2	1	0	1
E	1	1	2	1	0

Example shortest paths:

A->B: A-B (hops=1)
A->C: A-C (hops=1)
A->D: A-D (hops=1)
A->E: A-E (hops=1)

B->A: B-A (hops=1)
B->C: B-A-C (hops=2)
B->D: B-E-D (hops=2)
B->E: B-E (hops=1)

C->A: C-A (hops=1)
C->B: C-A-B (hops=2)
C->D: C-D (hops=1)
C->E: C-A-E (hops=2)

C->E: C-D-E (hops=2)

D->A: D-A (hops=1)
D->B: D-E-B (hops=2)
D->C: D-C (hops=1)
D->E: D-E (hops=1)

E->A: E-A (hops=1)
E->B: E-B (hops=1)
E->C: E-D-C (hops=2)
E->D: E-D (hops=1)

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> |

https://colab.research.google.com/drive/1D_VQDtRjX06aN03sNS46AP0lWzR-zJPR?usp=sharing

REFERENCES

GeeksforGeeks. (2025, August 27). *KMP algorithm for pattern searching*. GeeksforGeeks.

<https://www.geeksforgeeks.org/dsa/kmp-algorithm-for-pattern-searching/>

GeeksforGeeks. (2025, August 5). *Z algorithm (Linear time pattern searching Algorithm)*.

GeeksforGeeks.

<https://www.geeksforgeeks.org/dsa/z-algorithm-linear-time-pattern-searching-algorithm/>

GeeksforGeeks. (2024, April 20). *Naive algorithm for Pattern Searching*. GeeksforGeeks.

<https://www.geeksforgeeks.org/dsa/naive-algorithm-for-pattern-searching/>