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

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

HW10_BnBWP.py

Homework10Bn8_WP > HW10_BnBWP.py > ...

```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Luis Salomón Flores Ugalde

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ------ Homework 10 B&B and Welsh-Powell ------
from collections import deque

# ---- Build Graph ----
V = ["A", "B", "C", "D", "E"]
edges = {
    ("A", "B"),
    ("A", "C"),
    ("A", "D"),
    ("A", "E"),
    ("B", "E"),
    ("C", "D"),
    ("D", "E"),
}

def build_adj(vertices, edge_set):
    adj = {v: set() for v in vertices}
    for u, v in edge_set:
        adj[u].add(v)
        adj[v].add(u)
    return adj

ADJ = build_adj(V, edges)

# ---- Welsh-Powell Coloring (greedy by degree) ----
def welsh_powell_coloring(adj):
    order = sorted(adj, key=lambda v: len(adj[v]), reverse=True)
    colors = {}
    color_id = 0
    for v in order:
        if v in colors:
            continue
        colors[v] = color_id
        for u in order:
```

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 Algori
thms/Homework10Bn8_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

Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

HW10_BnBWP.py

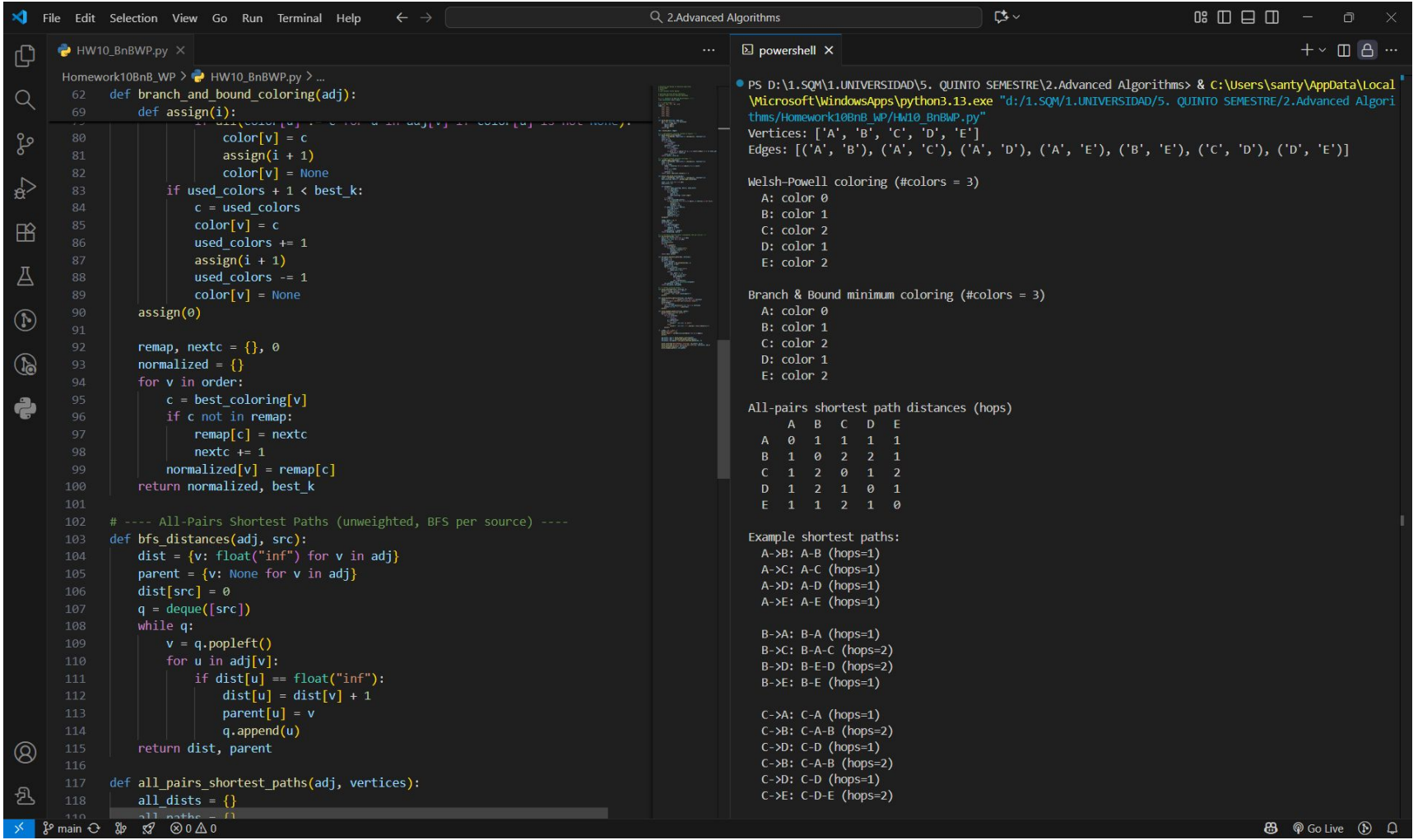
Homework10Bn8_WP > HW10_BnBWP.py > ...

```python
def welsh_powell_coloring(adj):
            continue
        colors[v] = color_id
        for u in order:
            if u in colors:
                continue
            if all(u not in adj[w] for w, c in colors.items() if c == color_i
                colors[u] = color_id
        color_id += 1
    return colors, color_id

# ---- Branch & Bound (minimum coloring) ----
def _greedy_upper_bound(adj):
    order = sorted(adj, key=lambda v: len(adj[v]), reverse=True)
    color = {}
    for v in order:
        taken = {color[u] for u in adj[v] if u in color}
        c = 0
        while c in taken:
            c += 1
        color[v] = c
    return color, max(color.values()) + 1

def branch_and_bound_coloring(adj):
    order = sorted(adj, key=lambda v: len(adj[v]), reverse=True)
    best_coloring, best_k = _greedy_upper_bound(adj)

    color = {v: None for v in adj}
    used_colors = 0

    def assign(i):
        nonlocal best_coloring, best_k, used_colors
        if i == len(order):
            k = used_colors
            if k < best_k:
                best_k = k
                best_coloring = color.copy()
            return
        v = order[i]
        for c in range(used_colors):
            if all(color[u] != c for u in adj[v] if color[u] is not None):
                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 Algori
thms/Homework10Bn8_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 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
HW10_BnBWP.py
Homework10Bn8_WP > HW10_BnBWP.py > ...
def branch_and_bound_coloring(adj):
def assign(i):
color[v] = c
assign(i + 1)
color[v] = None
if used_colors + 1 < best_k:
c = used_colors
color[v] = c
used_colors += 1
assign(i + 1)
used_colors -= 1
color[v] = None
assign(0)
remap, nextc = {}, 0
normalized = {}
for v in order:
c = best_coloring[v]
if c not in remap:
remap[c] = nextc
nextc += 1
normalized[v] = remap[c]
return normalized, best_k
# ---- All-Pairs Shortest Paths (unweighted, BFS per source) ----
def bfs_distances(adj, src):
dist = {v: float("inf") for v in adj}
parent = {v: None for v in adj}
dist[src] = 0
q = deque([src])
while q:
v = q.popleft()
for u in adj[v]:
if dist[u] == float("inf"):
dist[u] = dist[v] + 1
parent[u] = v
q.append(u)
return dist, parent
def all_pairs_shortest_paths(adj, vertices):
all_dists = {}
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/Homework10Bn8_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
Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

HW10_BnBWP.py

Homework10Bn8_WP > HW10_BnBWP.py > ...

```python
def all_pairs_shortest_paths(adj, vertices):
    all_dists = {}
    all_paths = {}
    for s in vertices:
        dist, parent = bfs_distances(adj, s)
        all_dists[s] = dist
        paths_s = {}
        for t in vertices:
            if dist[t] == float("inf"):
                paths_s[t] = None
            else:
                cur, path = t, []
                while cur is not None:
                    path.append(cur)
                    if cur == s:
                        break
                    cur = parent[cur]
                paths_s[t] = list(reversed(path))
        all_paths[s] = paths_s
    return all_dists, all_paths

# ---- Pretty printing helpers ----
def print_coloring(title, coloring, k):
    print(f"{title} (#colors = {k})")
    for v in sorted(coloring):
        print(f"  {v}: color {coloring[v]}")
    print()

def print_distance_matrix(vertices, all_dists):
    header = "    " + " ".join(f"{v:>3}" for v in vertices)
    print("All-pairs shortest path distances (hops)")
    print(header)
    for r in vertices:
        row = [f"{all_dists[c][r]:>3}" for c in vertices]
        print(f"{r:>3} " + " ".join(row))
    print()

def print_example_paths(vertices, paths):
    print("Example shortest paths:")
    for s in vertices:
        for t in vertices:
            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 Algori
thms/Homework10Bn8_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

Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

HW10_BnBWP.py

Homework10Bn8_WP > HW10_BnBWP.py > ...

```python
def print_example_paths(vertices, paths):
    for s in vertices:
        for t in vertices:
            if s == t:
                continue
            p = paths[s][t]
            if p is None:
                print(f"  {s}->{t}: no path")
            else:
                print(f"  {s}->{t}: {'-'.join(p)} (hops={len(p)-1})")
        print()

if __name__ == "__main__":
    print("Vertices:", V)
    print("Edges:", sorted(tuple(sorted(e)) for e in edges))
    print()

    wp_colors, wp_k = welsh_powell_coloring(ADJ)
    bb_colors, bb_k = branch_and_bound_coloring(ADJ)
    all_dists, all_paths = all_pairs_shortest_paths(ADJ, V)

    print_coloring("Welsh-Powell coloring", wp_colors, wp_k)
    print_coloring("Branch & Bound minimum coloring", bb_colors, bb_k)
    print_distance_matrix(V, all_dists)
    print_example_paths(V, 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/Homework10Bn8_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

Go Live

```
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)

  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/