

## Lab 11

COMP9021, Session 2, 2017

### 1 A generalised priority queue

Write a program `generalised_priority_queue_adt.py` that modifies `priority_queue_adt.py` so as to insert pairs of the form `(datum, priority)`. If a pair is inserted with a datum that already occurs in the priority queue, then the priority is (possibly) changed to the (possibly) new value. Next is a possible interaction.

```
$ python3
...
>>> from generalised_priority_queue_adt import *
>>> pq = PriorityQueue()
>>> L = [('A', 13), ('B', 13), ('C', 4), ('D', 15), ('E', 9), ('F', 4), ('G', 5),
...      ('H', 14), ('A', 4), ('B', 11), ('C', 15), ('D', 2), ('E', 17), ('A', 8),
...      ('B', 14), ('C', 12), ('D', 9), ('E', 5), ('A', 6), ('B', 16)]
>>> for e in L: pq.insert(e); print(f'pq._data[: len(pq) + 1]    len(pq._data)')
...
[None, ['A', 13]]      4
[None, ['A', 13], ['B', 13]]      4
[None, ['A', 13], ['B', 13], ['C', 4]]      4
[None, ['D', 15], ['A', 13], ['C', 4], ['B', 13]]      8
[None, ['D', 15], ['A', 13], ['C', 4], ['B', 13], ['E', 9]]      8
[None, ['D', 15], ['A', 13], ['C', 4], ['B', 13], ['E', 9], ['F', 4]]      8
[None, ['D', 15], ['A', 13], ['G', 5], ['B', 13], ['E', 9], ['F', 4], ['C', 4]]      8
[None, ['D', 15], ['H', 14], ['G', 5], ['A', 13], ['E', 9], ['F', 4], ['C', 4], ['B', 13]]      16
[None, ['D', 15], ['H', 14], ['G', 5], ['B', 13], ['E', 9], ['F', 4], ['C', 4], ['A', 4]]      16
[None, ['D', 15], ['H', 14], ['G', 5], ['B', 11], ['E', 9], ['F', 4], ['C', 4], ['A', 4]]      16
[None, ['D', 15], ['H', 14], ['C', 15], ['B', 11], ['E', 9], ['F', 4], ['G', 5], ['A', 4]]      16
[None, ['C', 15], ['H', 14], ['G', 5], ['B', 11], ['E', 9], ['F', 4], ['D', 2], ['A', 4]]      16
[None, ['E', 17], ['C', 15], ['G', 5], ['B', 11], ['H', 14], ['F', 4], ['D', 2], ['A', 4]]      16
[None, ['E', 17], ['C', 15], ['G', 5], ['B', 11], ['H', 14], ['F', 4], ['D', 2], ['A', 8]]      16
[None, ['E', 17], ['C', 15], ['G', 5], ['B', 14], ['H', 14], ['F', 4], ['D', 2], ['A', 8]]      16
[None, ['E', 17], ['B', 14], ['G', 5], ['C', 12], ['H', 14], ['F', 4], ['D', 2], ['A', 8]]      16
[None, ['E', 17], ['B', 14], ['D', 9], ['C', 12], ['H', 14], ['F', 4], ['G', 5], ['A', 8]]      16
[None, ['B', 14], ['H', 14], ['D', 9], ['C', 12], ['E', 5], ['F', 4], ['G', 5], ['A', 8]]      16
[None, ['B', 14], ['H', 14], ['D', 9], ['C', 12], ['E', 5], ['F', 4], ['G', 5], ['A', 6]]      16
[None, ['B', 16], ['H', 14], ['D', 9], ['C', 12], ['E', 5], ['F', 4], ['G', 5], ['A', 6]]      16
```

```

>>> for _ in range(len(pq)):
...     print(f'pq.delete():2 pq._data[: len(pq) + 1]    len(pq._data)')
...
B [None, ['H', 14], ['C', 12], ['D', 9], ['A', 6], ['E', 5], ['F', 4], ['G', 5]]    16
H [None, ['C', 12], ['A', 6], ['D', 9], ['G', 5], ['E', 5], ['F', 4]]    16
C [None, ['D', 9], ['A', 6], ['F', 4], ['G', 5], ['E', 5]]    16
D [None, ['A', 6], ['E', 5], ['F', 4], ['G', 5]]    8
A [None, ['G', 5], ['E', 5], ['F', 4]]    8
G [None, ['E', 5], ['F', 4]]    8
E [None, ['F', 4]]    8
F [None]    8

```

## 2 The Word Search puzzle

The Word Search puzzle consists of a grid of letters and a number of words, that have to be read horizontally, vertically or diagonally, in either direction. Write a program `word_search.py` that defines a class `WordSearch` with the following properties.

- To create a `WordSearch` object, the name of a file has to be provided. This file is meant to store a number of lines all with the same number of uppercase letters, those lines possibly containing spaces anywhere, and the file possibly containing extra blank lines.
- `__str__()` is implemented.
- It has a method `number_of_solutions()` to display the number of solutions for each word length for which a solution exists.
- It has a method `locate_word_in_grid()` that takes a word as argument; it returns `None` if the word cannot be read in the grid, and otherwise returns the  $x$  and  $y$  coordinates of an occurrence of the first letter of the word in the grid and the direction to follow (N, NE, E, SE, S, SW, W or NW) to read the whole word from that point onwards. Coordinates start from 0, with the x-axis pointing East, and the y-axis pointing South.
- It has a method `locate_words_in_grid()` that takes any number of words as arguments, and returns a dictionary whose keys are those words and whose values are `None` or the triple returned by `locate_word_in_grid()` when called with that word as argument.
- It has a method `display_word_in_grid()` that takes a word as argument and in case the word can be read from the grid, prints out the grid with all characters being displayed in lowercase, except for those that make up word, displayed in uppercase.

Here is a possible interaction.

```
$ cat word_search_1.txt
N D A O E L D L O G B M N E
I T D C M E A I N R U T S L
C L U U E I C G G G O L I I
K M U I M U I D I R I A L T
E U R T U N G S T E N B V H
L I L S L T T U L R U O E I
C M A T E T I U R D R C R U
I D S C A M A G N E S I U M
M A M P D M U I N A T I T I
P C N P L A T I N U M D L L
H Z E M A N G A N E S E I G
M G I T I N R U N O R I T C
R I A N N A M E R C U R Y N
U O T C C R E P P O C E E R
```

```

$ python3
...
>>> from word_search import *
>>> import pprint
>>> ws = WordSearch('word_search_1.txt')
>>> print(ws)
N D A O E L D L O G B M N E
I T D C M E A I N R U T S L
C L U U E I C G G G O L I I
K M U I M U I D I R I A L T
E U R T U N G S T E N B V H
L I L S L T T U L R U O E I
C M A T E T I U R D R C R U
I D S C A M A G N E S I U M
M A M P D M U I N A T I T I
P C N P L A T I N U M D L L
H Z E M A N G A N E S E I G
M G I T I N R U N O R I T C
R I A N N A M E R C U R Y N
U O T C C R E P P O C E E R
>>> metal = 'PLATINUM'
>>> print(f'{metal}: ws.locate_word_in_grid(metal)')
PLATINUM: (3, 9, 'E')
>>> metal = 'SODIUM'
>>> print(f'{metal}: ws.locate_word_in_grid(metal)')
SODIUM: None
>>> metals = ('PLATINUM', 'COPPER', 'MERCURY', 'TUNGSTEN', 'MAGNESIUM', 'ZINC', 'MANGANESE',
... 'TITANIUM', 'TIN', 'IRON', 'LITHIUM', 'CADMIUM', 'GOLD', 'COBALT', 'SILVER',
... 'NICKEL', 'LEAD', 'IRIDIUM', 'URANIUM', 'SODIUM')
>>> located_metals = ws.locate_words_in_grid(*metals)
>>> pprint.pprint(located_metals)
{'CADMIUM': (1, 9, 'N'),
 'COBALT': (11, 6, 'N'),
 'COPPER': (10, 13, 'W'),
 'GOLD': (9, 0, 'W'),
 'IRIDIUM': (10, 3, 'W'),
 'IRON': (11, 11, 'W'),
 'LEAD': (4, 5, 'S'),
 'LITHIUM': (13, 1, 'S'),
 'MAGNESIUM': (5, 7, 'E'),
 'MANGANESE': (3, 10, 'E'),
 'MERCURY': (6, 12, 'E'),
 'NICKEL': (0, 0, 'S'),
 'PLATINUM': (3, 9, 'E'),
 'SILVER': (12, 1, 'S'),
 'SODIUM': None,
 'TIN': (6, 9, 'NE'),
 'TITANIUM': (12, 8, 'W'),
 'TUNGSTEN': (3, 4, 'E'),
 'URANIUM': None,
 'ZINC': (1, 10, 'SE')}
>>> for metal in metals:
...     print(metal, end = ':\n')
...     ws.display_word_in_grid(metal)
...     print()
...
PLATINUM:
n d a o e l d l o g b m n e
i t d c m e a i n r u t s l
c l u u e i c g g g o l i i
k m u i m u i d i r i a l t
e u r t u n g s t e n b v h
l i l s l t t u l r u o e i
c m a t e t i u r d r c r u
i d s c a m a g n e s i u m
m a m p d m u i n a t i t i
p c n P L A T I N U M d l l
h z e m a n g a n e s e i g
m g i t i n r u n o r i t c
r i a n n a m e r c u r y n
u o t c c R E P P O C e e r

COPPER:
n d a o e l d l o g b m n e
i t d c m e a i n r u t s l
c l u u e i c g g g o l i i
k m u i m u i d i r i a l t
e u r t u n g s t e n b v h
l i l s l t t u l r u o e i
c m a t e t i u r d r c r u
i d s c a m a g n e s i u m
m a m p d m u i n a t i t i
p c n p l a t i n u m d l l
h z e m a n g a n e s e i g
m g i t i n r u n o r i t c
r i a n n a m e r c u r y n
u o t c c R E P P O C e e r

MERCURY:
n d a o e l d l o g b m n e
i t d c m e a i n r u t s l
c l u u e i c g g g o l i i

```

kmuiuidirialt  
eurtungstenbvh  
lilslttulruoei  
cmattetieurdreru  
idscamagnesium  
mampdmuinatiti  
pcnplatinumdll  
hzemanganeseig  
mgitinrunoritc  
riannaMERCURYn  
uotcreppoceer

TUNGSTEN:

ndaoelddlogbmne  
itdcmeainrutsll  
cluueicgggoli  
kmuiuidirialt  
eurTUNGSTENbvh  
lilslttulruoei  
cmattetieurdreru  
idscamagnesium  
mampdmuinatiti  
pcnplatinumdll  
hzemanganeseig  
mgitinrunoritc  
riannamercury  
uotcreppoceer

MAGNESIUM:

ndaoelddlogbmne  
itdcmeainrutsll  
cluueicgggoli  
kmuiuidirialt  
eurtungstenbvh  
lilslttulruoei  
cmattetieurdreru  
idscamagnesium  
mampdmuinatiti  
pcnplatinumdll  
hzemanganeseig  
mgitinrunoritc  
riannamercury  
uotcreppoceer

ZINC:

ndaoelddlogbmne  
itdcmeainrutsll  
cluueicgggoli  
kmuiuidirialt  
eurtungstenbvh  
lilslttulruoei  
cmattetieurdreru  
idscamagnesium  
mampdmuinatiti  
pcnplatinumdll  
hZemanganeseig  
mgItinrunoritc  
riaNnamercury  
uotcCreppoceer

MANGANESE:

ndaoelddlogbmne  
itdcmeainrutsll  
cluueicgggoli  
kmuiuidirialt  
eurtungstenbvh  
lilslttulruoei  
cmattetieurdreru  
idscamagnesium  
mampdmuinatiti  
pcnplatinumdll  
hzeMANGANESEig  
mgitinrunoritc  
riannamercury  
uotcreppoceer

TITANIUM:

ndaoelddlogbmne  
itdcmeainrutsll  
cluueicgggoli  
kmuiuidirialt  
eurtungstenbvh  
lilslttulruoei  
cmattetieurdreru  
idscamagnesium  
mampdMUINATITi  
pcnplatinumdll  
hzemanganeseig  
mgitinrunoritc  
riannamercury  
uotcreppoceer

TIN:

ndaoelddlogbmne  
itdcmeainruts1  
cluueicgggoli1  
kmuimuidirialt  
eurtungstenbvh  
lilsltttulruoei  
cmatetiurdrclu  
idscamagNesium  
mampdmuInatiti  
pcnplaTinumd11  
hzemanganeseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

IRON:

ndaoelddlogbmne  
itdcmeainruts1  
cluueicgggoli1  
kmuimuidirialt  
eurtungstenbvh  
lilsltttulruoei  
cmatetiurdrclu  
idscamagnesium  
mampdmuInatiti  
pcnplatinumd11  
hzemanganeseig  
mgitinrunORItc  
riannamercuryn  
uotccreppoceer

LITHIUM:

ndaoelddlogbmne  
itdcmeainrutsL  
cluueicgggoliI  
kmuimuidirialT  
eurtungstenbvh  
lilsltttulruoeI  
cmatetiurdrclU  
idscamagnesium  
mampdmuInatiti  
pcnplatinumd11  
hzemanganeseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

CADMIUM:

ndaoelddlogbmne  
itdcmeainruts1  
cluueicgggoli1  
kMuimuidirialt  
eUrtungstenbvh  
lIlsltttulruoei  
cMatetiurdrclu  
iDscamagnesium  
mAmpdmuInatiti  
pCnplatinumd11  
hzemanganeseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

GOLD:

ndaoelDLOGbmne  
itdcmeainruts1  
cluueicgggoli1  
kmuimuidirialt  
eurtungstenbvh  
lilsltttulruoei  
cmatetiurdrclu  
idscamagnesium  
mampdmuInatiti  
pcnplatinumd11  
hzemanganeseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

COBALT:

ndaoelddlogbmne  
itdcmeainruTsl  
cluueicgggoLi1  
kmuimuidiriAlt  
eurtungstenBvh  
lilsltttulruOei  
cmatetiurdrCru  
idscamagnesium  
mampdmuInatiti  
pcnplatinumd11  
hzemanganeseig

mgitinrunoritc  
riannamercuryn  
uotccreppoceer

SILVER:

ndaoelddlogbmne  
itdcmeainrutsSl  
cluueicgggoliIi  
kmuiuidirialt  
eurtungstenbVh  
lilslttulruoEi  
cmattetiurdrCRu  
idscamagnesium  
mampdmuinatititi  
pcnplatinumdll  
hzemanganesseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

NICKEL:

Ndaoelddlogbmne  
It dcmeainrutsSl  
CluueicgggoliIi  
Kmuimuidirialt  
Eurtungstenbvh  
Lilslttulruoei  
cmattetiurdrCRu  
idscamagnesium  
mampdmuinatititi  
pcnplatinumdll  
hzemanganesseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

LEAD:

ndaoelddlogbmne  
itdcmeainrutsSl  
cluueicgggoliIi  
kmuiuidirialt  
eurtungstenbvh  
lilslttulruoei  
cmatEtiurdrCRu  
idscAmagnesium  
mampDmuinatititi  
pcnplatinumdll  
hzemanganesseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

IRIDIUM:

ndaoelddlogbmne  
itdcmeainrutsSl  
cluueicgggoliIi  
kmuiMUIDIRIalt  
eurtungstenbvh  
lilslttulruoei  
cmattetiurdrCRu  
idscamagnesium  
mampdmuinatititi  
pcnplatinumdll  
hzemanganesseig  
mgitinrunoritc  
riannamercuryn  
uotccreppoceer

URANIUM:

SODIUM:

### 3 Median and priority queues (optional)

Write a program `median.py` that implement a class `Median` that allows one to manage a list  $L$  of values with the following operations:

- add a value in logarithmic time complexity;
- return the median in constant time complexity.

One possible design is to use two priority queues: a max priority queue to store the half of the smallest elements, and a min priority queue to store the half of the largest elements. Both priority queues have the same number of elements if the number of elements in  $L$  is even, in which case the median is the average of the elements at the top of both priority queues. Otherwise, one priority queue has one more element than the other, and its element at the top is the median. For the priority queue interface, extend `priority_queue_adt.py` to `max_or_min_priority_queue_adt.py`, adapting `PriorityQueue` and adding two classes, namely, `MaxPriorityQueue` and `MinPriorityQueue`, that both inherit from `PriorityQueue`, and allow for the appropriate comparisons to be performed. Also add in `PriorityQueue` a method `top_priority()` to return the value at the top of the priority queue (`None` in case the priority queue is empty).

Next is a possible interaction.

```
$ python3
...
>>> from max_or_min_priority_queue_adt import *
>>> max_pq = MaxPriorityQueue()
>>> min_pq = MinPriorityQueue()
>>> L = [13, 13, 4, 15, 9, 4, 5, 14, 4, 11, 15, 2, 17, 8, 14, 12, 9, 5, 6, 16]
>>> for e in L:
...     max_pq.insert(e)
...     min_pq.insert(e)
...
>>> max_pq._data[: len(max_pq) + 1]
[None, 17, 16, 15, 13, 15, 5, 14, 13, 6, 14, 11, 2, 4, 4, 8, 12, 9, 4, 5, 9]
>>> max_pq.top_priority()
17
>>> min_pq._data[: len(min_pq) + 1]
[None, 2, 4, 4, 5, 11, 4, 5, 9, 6, 13, 15, 13, 17, 8, 14, 15, 12, 14, 9, 16]
>>> min_pq.top_priority()
2
```



```

>>> for i in range(len(max_pq) // 2): print(max_pq.delete())
...
17
16
15
15
14
14
13
13
12
11
>>> for i in range(len(min_pq) // 2): print(min_pq.delete())
...
2
4
4
4
5
5
6
8
9
9
>>> from median import *
>>> L = [13, 13, 4, 15, 9, 4, 5, 14, 4, 11, 15, 2, 17, 8, 14, 12, 9, 5, 6, 16]
>>> values = Median()
>>> for e in L: values.insert(e); values.median()
...
13
13.0
13
13.0
13
11.0
9
11.0
9
10.0
11
10.0
11
10.0
11
11.5
11
10.0
9
10.0

```

## 4 Voting systems (optional)

Find out (e.g., in Wikipedia) about these voting systems: (a) one round method, (b) two round method, (c) elimination method, (d) De Borda count, and (e) De Condorcet count. Then design a program `election.py` that defines a class `Election`, with objects of this class created from Excel files of the kind provided as examples, to which the methods

- `one_round_winners()`,
- `two_round_winners()`,
- `elimination_winner()`,
- `de_borda_winners()`, and
- `de_condorcet_winners()`

can be applied. Also, the `__str__()` method is implemented so as to display in textual form the election results recorded in the Excel file.

Next is a possible interaction.

```
$ python3
...
>>> from election import *
>>> election = Election('election_1.xlsx')
>>> print(election)
Number of votes  Albert  Emily  Oscar  Maria  Max
3273             1      5      4      2      3
2182             5      1      4      3      2
1818             5      2      1      4      3
1636             5      4      2      1      3
727              5      2      4      3      1
364              5      4      2      3      1
>>> election.one_round_winners()
The winner is Albert.
>>> election.two_round_winners()
The winner is Emily.
>>> election.elimination_winners()
The winner is Oscar.
>>> election.de_borda_winners()
The winner is Maria.
>>> election.de_condorcet_winners()
The winner is Max.
```

```

>>> election = Election('election_2.xlsx')
Number of votes  Albert  Emily  Oscar  Maria  Max
               1000      1      2      3      4      5
>>> election.one_round_winners()
The winner is Albert.
>>> election.two_round_winners()
The winner is Albert.
>>> election.elimination_winners()
The winner is Max.
>>> election.de_borda_winners()
The winner is Albert.
>>> election.de_condorcet_winners()
The winner is Albert.
>>> election = Election('election_3.xlsx')
>>> print(election)
Number of votes  Albert
               1000      1
               1000      1
               1000      1
               1000      1
               1000      1
               1000      1
>>> election.one_round_winners()
All candidates are winners.
>>> election.two_round_winners()
All candidates are winners.
>>> election.elimination_winners()
All candidates are winners.
>>> election.de_borda_winners()
All candidates are winners.
>>> election.de_condorcet_winners()
All candidates are winners.
>>> election = Election('election_4.xlsx')
>>> print(election)
Number of votes  Albert  Emily  Oscar
               1000      1      2      3
               1000      2      1      3
>>> election.one_round_winners()
The winners are Albert and Emily
>>> election.two_round_winners()
The winners are Albert and Emily
>>> election.elimination_winners()
The winner is Oscar.
>>> election.de_borda_winners()
The winners are Albert and Emily
>>> election.de_condorcet_winners()
The winners are Albert and Emily

```

```

>>> election = Election('election_5.xlsx')
>>> print(election)
Number of votes  Albert  Emily  Oscar  Maria
1000             1       2       3       4
1000             2       3       1       4
1000             3       1       2       4
>>> election.one_round_winners()
The winners are Albert, Emily and Oscar
>>> election.two_round_winners()
The winners are Albert, Emily and Oscar
>>> election.elimination_winners()
The winner is Maria.
>>> election.de_borda_winners()
The winners are Albert, Emily and Oscar
>>> election.de_condorcet_winners()
There is no winner.
>>> election = Election('election_6.xlsx')
>>> print(election)
Number of votes  Albert  Emily  Oscar
1000             1       2       3
1000             2       1       3
250              2       3       1
250              3       2       1
>>> election.one_round_winners()
The winners are Albert and Emily
>>> election.two_round_winners()
The winners are Albert and Emily
>>> election.elimination_winners()
The winners are Albert and Emily
>>> election.de_borda_winners()
The winners are Albert and Emily
>>> election.de_condorcet_winners()
The winners are Albert and Emily

```