Assignment 5

1) The translated octrees of object B after the x-axis translation and after the final y-axis translation respectively in the linear form.

⟹ ·) order of translation is +5 units along the negative x-axis

→ 2's complement of 5 units along the negative x-axis will be ⟹ 11111011

·) order of translation is 48 units along the positive y-axis

→ Binary form of 48 units along the positive y-axis will be ⟹ 00110000

Manual calculations have been performed for the first 54 cells in the object B to verify that the code works well for the translations.

Translated linear octrees for the object B have been calculated using the table given in the paper by Ahuja and Nash.

2) After translation, all of the new Object B octrees are fully expanded to a depth of 9 (8 digits): rXXXXXXXX. Translated Object B can be referred to as Object B' (B' = B prime).

Since Object A + Object B' are expanded to the same depth, finding the intersection is as simple as finding sequences in Object B' that are identical to sequences in Object A. In Python it was simply converting the lists to hashable objects and using the intersection operator (&) which compares the hashes of the sequences in both objects. If the hashes are identical, there is an intersection.

If there is an intersection, the volume can be found by multiplying the cubed length of the node by the number of intersecting nodes. The length of the nodes is determined by successively halving the parents length. For our case, the length of the nodes being compared was $100/2^8 = .391\text{cm}$. This means the volume is $m * \left(100/2^8\right)^3$ where m is the number of intersection nodes. In our problem, there were zero intersections making $m = 0$ and the intersecting volume also zero ($V = 0$).

(1) Manual calculations for the translations of the linear octrees

1) γ52000120
   11111011

γ43111131
00110000

γ43331131

2) γ52000121
   11111011

γ52000020
00110000

γ5222002 0

3) γ52000122
   11111011

γ43111133
00110000

γ43331133

4) γ52000123
   11111011

γ52000002 2
00110000

γ52222002 2

5) γ52000124
   11111011

γ43111135
00110000

γ43331135

6) γ52000125
   ϕϕ111011

52000024
00110000

γ52222 0024

(2)

7) ٢52000126
1111101 1

٢ 43111137
00110000

٢ 4333 1137

8) ٢52000127
1111101 1

٢52000026
00110000

٢ 52220026

1) ٢52000130
11111011

٢ 52000021
00110000

٢ 52220021

2) ٢52000131
11111011

٢52000030
00110000

٢52220030

3) ٢52000132
11111011

٢52000023
00110000

٢52220023

4) ٢52000133
11111011

٢52000032
00110000

٢52220032

(3)

5) 1 1 1 1 1 1 1
γ52000134
1 1 1 1 1 1 0 1 1

γ52000025
00110000

γ52220025

6) 1 1 1 1 1 1 1 1
γ52000135
1 1 1 1 1 1 0 1 1

γ52000034
00110000

γ52220034

7) 1 1 1 1 1 1 1
γ52000136
1 1 1 1 1 1 0 1 1

γ52000027
00110000

γ52220027

8) 1 1 1 1 1 1 1
γ52000137
1 1 1 1 1 1 0 1 1

γ52000036
00110000

γ52220036

1) 1 1 1 1 1 1 1 1
γ52000160
1 1 1 1 1 1 0 1 1

γ43111171
00110000

γ43331171

2) 1 1 1 1 1 1 1 1
γ52000161
1 1 1 1 1 1 0 1 1

γ52000060
00110000

γ52220060

(4)

3) 52000162
1111 1011

43111173
00110000

43331173

4) 52000163
1111 1011

52000062
00110000

52220062

5) 52000164
1111 1011

43111175
00110000

43331175

6) 52000165
1111 1011

52000064
00110000

52220064

7) 52000166
1111 1011

43111177
00110000

43331177

8) 52000167
1111 1011

52000066
00110000

52220066

(5)

classmate
Date
Page

1) 0x52000170
11111011

0x52000061
00110000

0x52200061

4) 0x52000173
11111011

0x52000072
00110000

0x52200072

2) 0x52000171
11111011

0x52000070
00110000

0x52200070

5) 0x52000174
11111011

0x52000065
00110000

0x52200065

3) 0x52000172
11111011

0x52000063
00110000

0x52200063

6) 0x52000175
11111011

0x52000074
00110000

0x52200074

7) 0x52000176
11111011

0x52000067
00110000

0x52200067

8) 0x52000177
11111011

0x52000076
00110000

0x52200076

1)
```
 γ52000520
  1111101 1
```
```
 γ43111531
  00110000
```
γ43331531

2)
```
   1111111
 γ52000521
  11111011
```
```
 γ52000420
  00110000
```
γ52220420

3)
```
 γ52000522
  11111011
```
```
 γ43111533
  00110000
```
γ43331533

4)
```
   1 1111111
 γ52000523
  11111011
```
```
 γ52000422
  00110000
```
γ52220422

1)
```
   1 1 1 1 1 1
 γ52000530
  11111011
```
```
 γ52000421
  00110000
```
γ52220421

2)
```
   1 1 1 1 1 1
 γ52000531
  11111011
```
```
 γ52000430
  00110000
```
γ52220430

3) 1 1 1 1 1 1 1

r 5 2 0 0 0 5 3 2

1 1 1 1 1 0 1 1

r 5 2 0 0 0 4 2 3

0 0 1 1 0 0 0 0

r 5 2 2 2 0 4 2 3

4) 1 1 1 1 1 1 1

r 5 2 0 0 0 5 3 3

1 1 1 1 1 0 1 1

r 5 2 0 0 0 4 3 2

0 0 1 1 0 0 0 0

r 5 2 2 2 0 4 3 2

1) r 5 2 0 2 2 3 0 0

1 1 1 1 1 0 1 1

1 1 1

r 4 3 1 3 3 3 1 1

0 0 1 1 0 0 0 0

r 6 1 1 1 3 3 1 1

2) 1 1 1 1 1 1 1

r 5 2 0 2 2 3 0 1

1 1 1 1 1 0 1 1

1 1 1

r 5 2 0 2 2 2 0 0

0 0 1 1 0 0 0 0

* **7 0 0 0 2 2 0 0**

3) r 5 2 0 2 2 3 0 2

1 1 1 1 1 0 1 1

1 1 1

r 4 3 1 3 3 3 1 3

0 0 1 1 0 0 0 0

r 6 1 1 1 3 3 1 3

4) 1 1 1 1 1 1 1

r 5 2 0 2 2 3 0 3

1 1 1 1 1 0 1 1

1 1 1

r 5 2 0 2 2 2 0 2

0 0 1 1 0 0 0 0

* **7 0 0 0 2 2 0 2**

(8)

classmate
Date
Page

5) r520 22304
   11111011
   r43 133315
   00110000
   r61 11315

8) r520 22307
   11111011
   r520 22206
   00110000
   r70002206 ☆

6) r520 22305
   11111011
   r520 22204
   00110000
   ☆ **r70002204**

1) r520 22310
   11111011
   r520 22201
   00110000
   r70002201 ☆

7) r520 22306
   11111011
   r43 133317
   00110000
   r61113317

2) r520 22311
   11111011
   r520 22210
   00110000
   r72002210 ☆

3) γ52022312
11111011

γ52022203
00110000

★ γ70002203

6) γ52022315
11111011

γ52022214
00110000

★ γ70002214

4) γ52022313
11111011

γ52022212
00110000

γ70002212 ★

7) γ52022316
11111011

γ52022207
00110000

70002207 ★

5) γ52022314
11111011

γ52022205
00110000

γ70002205 ★

8) γ52022317
11111011

γ52022216
00110000

γ70002216 ★

(7) Final Results for the first 54 octrees in the object B

1	γ52000120	⟹	γ43331131
2	γ52000121	⟹	γ52220020
3	γ52000122	⟹	γ43311133
4	γ52000123	⟹	γ52220022
5	γ52000124	⟹	γ43331135
6	γ52000125	⟹	γ52220024
7	γ52000126	⟹	γ43331137
8	γ52000127	⟹	γ52220026

1	γ52000130	⟹	γ52220021
2	γ52000131	⟹	γ52220030
3	γ52000132	⟹	γ52220023
4	γ52000133	⟹	γ52220032
5	γ52000134	⟹	γ52220025
6	γ52000135	⟹	γ52220034
7	γ52000136	⟹	γ52220027
8	γ52000137	⟹	γ52220036

1	γ52000160	⟹	γ43331171
2	γ52000161	⟹	γ52220060
3	γ52000162	⟹	γ43331173
4	γ52000163	⟹	γ52220062
5	γ52000164	⟹	γ43331175
6	γ52000165	⟹	γ52220064
7	γ52000166	⟹	γ43331177
8	γ52000167	⟹	γ52220066

(11) ~~Res~~

classmate Date Page

1. ₹52000170 ⟶ ₹52220061
2. ₹52000171 ⟶ ₹52220070
3. ₹52000172 ⟶ ₹52220063
4. ₹52000173 ⟶ ₹52220072
5. ₹52000174 ⟶ ₹52220065
6. ₹52000175 ⟶ ₹52220074
7. ₹52000176 ⟶ ₹52220067
8. ₹52000177 ⟶ ₹52220076

1. ₹52000520 ⟶ ₹43315311
2. ₹52000521 ⟶ ₹52204200
3. ₹52000522 ⟶ ₹43331533
4. ₹52000523 ⟶ ₹52220422

1. ₹52000530 ⟶ ₹52220421
2. ₹52000531 ⟶ ₹52220430
3. ₹52000532 ⟶ ₹52204233
4. ₹52000533 ⟶ ₹52204322

1. ₹52022300 ⟶ ₹61113311
2. ₹52022301 ⟶ ₹70002200 ✱
3. ₹52022302 ⟶ ₹61113313
4. ₹52022303 ⟶ ₹70002202 ✱
5. ₹52022304 ⟶ ₹61113315
6. ₹52022305 ⟶ ₹70002204 ✱
7. ₹52022306 ⟶ ₹61113317
8. ₹52022307 ⟶ ₹70002206 ✱

1	₹52022310	⟶	₹70002201 ✱
2	₹52022311	⟶	₹72002210 ✱
3	₹52022312	⟶	₹70002203 ✱
4	₹52022313	⟶	₹70002212 ✱
5	₹52022314	⟶	₹70002205 ✱
6	₹52022315	⟶	₹70002214 ✱
7	₹52022316	⟶	₹70002207 ✱
8	₹52022317	⟶	₹70002216 ✱

hw5

April 13, 2020

[9]:
```
!jupyter nbconvert --to script hw5.ipynb
!jupyter nbconvert --to pdf hw5.ipynb
```

```
[NbConvertApp] Converting notebook hw5.ipynb to script
[NbConvertApp] Writing 4174 bytes to hw5.py
[NbConvertApp] Converting notebook hw5.ipynb to pdf
[NbConvertApp] Writing 36261 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 35457 bytes to hw5.pdf
```

[13]:
```
from typing import *
import numpy as np
import bitstring as bs
```

[14]:
```
# max_len determines the maximum length in the second dimension of Python list
def max_len(items: List) -> int:
    maxLen = 0
    for item in items:
        if (itemLen := len(item)) > maxLen:
            maxLen = itemLen
    return maxLen

# uniform_octree_len expands octree levels to where the depths are uniform
def uniform_octree_len(rSequences: List, maxLen: int) -> List:
    shortLenMaskFunc = lambda rSeq: len(rSeq) != maxLen
    while any(shortLenMask := list(map(shortLenMaskFunc, rSequences))):
        i = shortLenMask.index(True)
        rSequences.extend([rSequences[i].copy() + [str(digit)] for digit in␣
 ↪range(8)])
        del rSequences[i]

    return sorted(rSequences)
```

```python
# lookup_table for Table 2 from Ahuja and Nash
def lookup_table(label: Union[str, int], direction: Union[str, int]) -> str:
    if direction not in {"x", "y", "z", "0", "1", "2", 0, 1, 2}:
        raise IndexError("direction must be in {'x', 'y', 'z', '0', '1', '2',␣
↪0, 1, 2}")
    if type(label) == str:
        label = int(label)
    if label > 7 or label < 0:
        raise IndexError("label must be between 0 and 7")

    if direction in ("x", "y", "z"):
        direction = ("x", "y", "z").index(direction)
    elif direction in ("0", "1", "2"):
        direction = int(direction)

    table = (
        (1, 2, 4),
        (10, 3, 5),
        (3, 10, 6),
        (12, 11, 7),
        (5, 6, 10),
        (14, 7, 11),
        (7, 14, 12),
        (16, 15, 13)
    )
    return str(table[label][direction])

# octree_displacement returns the rSequence after it has been translated in␣
↪`direction` by the `displacement` (binary/2's comp) using the `lookup_table`
def octree_displacement(rSeq: str, displacement: str, direction: Union[int,␣
↪str]) -> List:
    # copy so we don't mutate outside lists
    displacement = displacement.copy()
    rSeq = rSeq.copy()

    for i in reversed(range(len(displacement))):
        if int(displacement[i]) == 2:
            displacement[i] = str(0)
            if i != 0:
                displacement[i-1] = str(int(displacement[i-1]) + 1)

        if int(displacement[i]) == 0:
            continue

        tableVal = lookup_table(rSeq[i], direction)
        if int(tableVal) < 10:
            rSeq[i] = tableVal
```

```
        else:
            rSeq[i] = tableVal[-1]
            if i != 0:
                displacement[i-1] = str(int(displacement[i-1]) + 1)

    return rSeq
```

[15]:
```
# read in A and B from the text files (copied from .pdf document)
with open("ObjectA.txt", "rt") as f:
    A  = [rSeq.strip() for rSeq in f.read().split(",")]

with open("ObjectB.txt", "rt") as f:
    B  = [rSeq.strip() for rSeq in f.read().split(",")]

# split into letter lists & drop the 'r'
A = [list(rSeq)[1:] for rSeq in A]
B = [list(rSeq)[1:] for rSeq in B]

# find the length needed to expand to
if (ALen := max_len(A)) >= (BLen := max_len(B)):
    maxLen = ALen
elif (ALen := max_len(A)) < (BLen := max_len(B)):
    maxLen = BLen

# make uniform lengths
A = uniform_octree_len(A, maxLen)
B = uniform_octree_len(B, maxLen)

# x and y binary/2's comp representation
x = list(str(bs.Bits(int=-5, length=maxLen).bin))
y = list(str(bs.Bits(int=48, length=maxLen).bin))

# translate B
BPrime = []
for b in B:
    BPrimeX = octree_displacement(b, x, "x")
    BPrimeXY = octree_displacement(BPrimeX, y, "y")
    BPrime.append(BPrimeXY)

# test two results
assert("".join(BPrime[7]) == "52220026")
assert("".join(BPrime[-1]) == "70007036")

# write it out to file
print(f"B': ")
with open("BPrime.txt", "wt") as f:
    for b in BPrime:
```

```
        out = f"r{''.join(b)}"
        print(out)
        f.write(f"{out}\n")

# need to convert A and BPrime to hashable containers
A = set([tuple(a) for a in A])
BPrime = set([tuple(b) for b in BPrime])

# find intersection and volume
intersection = A & BPrime
volSmallestOctree = (100 / (2**maxLen) ** 3)
volTotal = len(intersection) * volSmallestOctree
print(f"Intersection volume: {volTotal} cm^3")
```

```
B':
r43331131
r52220020
r43331133
r52220022
r43331135
r52220024
r43331137
r52220026
r52220021
r52220030
r52220023
r52220032
r52220025
r52220034
r52220027
r52220036
r43331171
r52220060
r43331173
r52220062
r43331175
r52220064
r43331177
r52220066
r52220061
r52220070
r52220063
r52220072
r52220065
r52220074
r52220067
r52220076
```

r43331531
r52220420
r43331533
r52220422
r52220421
r52220430
r52220423
r52220432
r61113311
r70002200
r61113313
r70002202
r61113315
r70002204
r61113317
r70002206
r70002201
r70002210
r70002203
r70002212
r70002205
r70002214
r70002207
r70002216
r61113351
r70002240
r61113353
r70002242
r61113355
r70002244
r61113357
r70002246
r70002241
r70002250
r70002243
r70002252
r70002245
r70002254
r70002247
r70002256
r61113711
r70002600
r61113713
r70002602
r70006111
r70007000
r70006113
r70007002

```
r70006115
r70007004
r70006117
r70007006
r70007001
r70007010
r70007003
r70007012
r70007005
r70007014
r70007007
r70007016
r70006131
r70007020
r70006133
r70007022
r70006135
r70007024
r70006137
r70007026
r70007021
r70007030
r70007023
r70007032
r70007025
r70007034
r70007027
r70007036
Intersection volume: 0.0 cm^3
```

[ ]: