

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC FPT TP. HCM
CHUYÊN NGÀNH: TRÍ TUỆ NHÂN TẠO
MÔN HỌC: CSD203



GIẢI THUẬT DI TRUYỀN VỚI BÀI TOÁN
SẮP XẾP THỜI KHÓA BIỂU



FPT UNIVERSITY

Giảng viên: Bùi Thanh Hùng.

Sinh viên thực hiện:

+ Trần Xuân Thành: SE160972

+ Trương Tuấn Phi: SE161036

TP. HCM, tháng 7 năm 2022



PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

Thành phố Hồ Chí Minh, ngày tháng năm

Giảng viên đánh giá:

MỤC LỤC

Nội dung

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN	1
MỤC LỤC.....	2
LỜI NÓI ĐẦU	3
CHƯƠNG I. GIỚI THIỆU BÀI TOÁN.....	4
1. Phát biểu bài toán.....	4
2. Ý nghĩa bài toán.....	4
3. Đánh giá bài toán	4
4. Phạm vi giải quyết	5
CHƯƠNG II. GIẢI QUYẾT BÀI TOÁN.....	6
1. Cấu trúc dữ liệu.....	6
2. Sơ đồ giải thuật và hiện thực	10
3. Kết quả và thảo luận	23
TỔNG KẾT	24
TÀI LIỆU THAM KHẢO.....	25
TỰ ĐÁNH GIÁ.....	26

LỜI NÓI ĐẦU

Bài toán *Sắp xếp Thời khóa biểu* là một bài toán hay và có tính thực tiễn rất cao, tuy nhiên đây lại là một bài toán phức tạp với nhiều ràng buộc và sắp xếp trên không gian rộng lớn. Đặc biệt, đây cũng là một bài toán không cố định kết quả đầu ra, vậy nên việc ứng dụng giải thuật di truyền vào giải quyết bài toán là khá hợp lí. Giải thuật này đơn giản giúp ta có thể tìm ra các giải pháp tốt hơn hoặc tối ưu hơn các giải pháp hiện có ở thời điểm hiện tại. Sau cùng là tuân thủ được các ràng buộc mà chúng ta mong muốn.

Giải thuật di truyền mô phỏng lại quá trình tiến hóa của sinh vật thông qua quá trình lai tạo và chọn lọc tự nhiên. Từ đó tạo ra các cá thể con có bộ NST tốt hơn thế hệ cha mẹ của chúng (chưa thể là tốt nhất vì sau quá trình này, các thế hệ sau sẽ có bộ NST tiến hóa hơn các thế hệ trước).

Thông qua bài tập này, chúng em mong muốn học hỏi nhiều hơn và có thể ứng dụng những điều đã được học ở trường vào các bài tập ứng dụng thực tế như thế này. Vì là lần đầu tiên, chúng em rất mong được thầy góp ý và điều chỉnh sai sót để chúng em có thể tiếp thu và làm tốt hơn ở lần sau!

Nhóm sinh viên: Trương Tuấn Phi, Trần Xuân Thành – A11705.

CHƯƠNG I. GIỚI THIỆU BÀI TOÁN

1. Phát biểu bài toán

Cho các thông tin về thời khoá biểu ở một trường học:

- ❖ Giáo viên và môn học: giáo viên có thể dạy những môn nào.
- ❖ Giáo viên và số tiết học tối thiểu, tối đa trong một tuần.
- ❖ Phòng học, ca học: có những phòng học nào; mỗi phòng học trong 1 ngày được chia làm mấy ca, mỗi ca bao nhiêu tiết.
- ❖ Trong học kỳ này, mỗi môn học cần được tổ chức bao nhiêu lớp.

Bài toán: In ra thời khóa biểu cho từng giảng viên, từng lớp trong học kỳ (thời khóa biểu phải thỏa mãn các ràng buộc đã cho trước và khả thi trong thực tế, càng tạo thuận lợi cho công tác giảng dạy của giảng viên – sinh viên thì càng tốt).

2. Ý nghĩa bài toán

Bài toán có ý nghĩa thực tiễn rất cao. Giúp cung cấp thông tin giảng dạy và học tập cho giảng viên – sinh viên một cách nhanh chóng, tiện lợi và chính xác...

3. Đánh giá bài toán

- ❖ Đây là một bài toán không có một kết quả cụ thể/đồng nhất trong mỗi lần giải (multi-solution) → Không xác định chi tiết đầu ra.
- ❖ Dễ gây xung đột/ mâu thuẫn vì chứa nhiều ràng buộc.

- ❖ Phát sinh nhiều biến cố trong lúc vận hành (chẳng hạn: Giảng viên bị ốm, phòng học bị hư hỏng, ứng phó với COVID-19,...).
- ❖ Tổng hợp, phân loại, truy xuất dữ liệu phức tạp, nhiều bước (đa tầng/chồng chéo).

4. Phạm vi giải quyết

Vì đây là một đề tài phức tạp, đòi hỏi khoảng thời gian nghiên cứu lâu dài, vận dụng nhiều kiến thức chuyên ngành nên trong bài tập này chúng em xin được giải quyết việc phân chia công việc sao cho phù hợp các tiêu chí đề ra trước đó. Sau cùng, in ra thời khóa biểu cho từng giảng viên và từng lớp với độ tối ưu đáng kể.

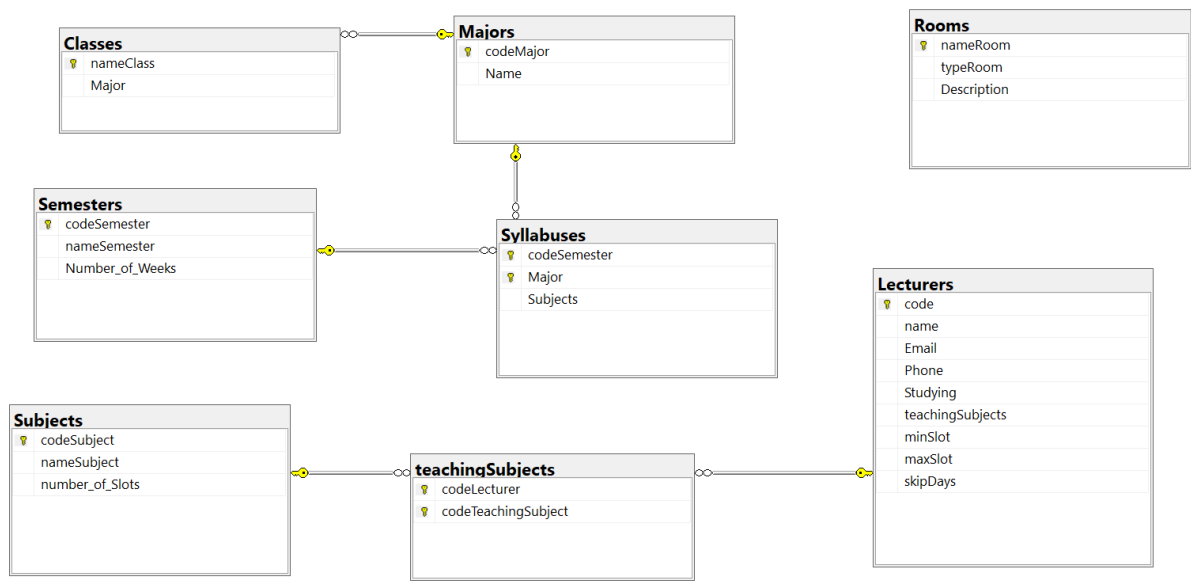
CHƯƠNG II.

GIẢI QUYẾT BÀI TOÁN

1. Cấu trúc dữ liệu

- DỮ LIỆU ĐẦU VÀO:

Bài toán nhận dữ liệu đầu vào gồm các thông tin về giảng – dạy từ một cơ sở dữ liệu chứa các bảng với sự liên kết như mô hình sau:




Hình 1. Mô hình liên kết giữa các bảng trong Cơ sở dữ liệu đầu vào

✚ Một số bảng dữ liệu chính:

- ❖ Bảng **Classes**: gồm thông tin về các lớp và chuyên ngành học của từng lớp.
- ❖ Bảng **Lecturers**: gồm thông tin chi tiết về các giảng viên (trong đó bao gồm thông tin về các môn giảng viên dạy, số slot yêu cầu dạy tối thiểu

trong tuần, số slot yêu cầu dạy tối đa trong tuần và các ngày nghỉ trong tuần đối với từng giảng viên,...)

- ❖ Bảng **Subjects**: thông tin về các môn học và số slot học theo chương trình của môn học đó.
- ❖ Bảng **Majors**: thông tin về tên các chuyên ngành.
- ❖ Bảng **Semesters**: thông tin về các học kỳ và số tuần học của từng học kỳ.
- ❖ Bảng **Syllabuses**: thông tin chi tiết về các môn cần học của mỗi chuyên ngành ở từng học kỳ.
- ❖ Bảng **Rooms**: thông tin về các phòng học.

 Các bảng dữ liệu mẫu:

Class	Major
AI1071	AI
SE1001	SE
SE1011	SE
AI1072	AI
AI1073	AI
AI1074	AI

Major	Name
AI	Artificial Intelligence
SE	Software Engineering

Hình 2. Dữ liệu mẫu của bảng *Classes*

Hình 3. Dữ liệu mẫu của bảng *Majors*

code	name	Email	Phone	Studying	teachingSubjects	minSlot	maxSlot	skipDays
VinhNT01	Nguyễn Thế Vinh	vinhnt01@fe.edu.vn	NULL	Computer Science	MAE101,AIG201c,CSD203	7	24	NULL
BaoTG02	Trương Gia Bảo	baotg02@fe.edu.vn	0936759826	Data Science	MAD101,AIG201c,CSD203,DBI202	9	24	Wed, Sat
BinhPT01	Phạm Thị Bình	binhpt01@fe.edu.vn	NULL	Software Engineer	MAE101,MAD201,TRS6	8	24	NULL
KiemHH02	Hồ Hoàn Kiếm	kiemhh02@fe.edu.vn	NULL	Software Engineer	PFP203,CEA201,CSI104,PRF107	12	24	NULL
PhiTT	Trương Tuấn Phi	NULL	NULL	NULL	CSD203	20	30	NULL
ThanhTX	TRAN XUAN THANH	NULL	NULL	NULL	CSD203	0	50	NULL

Hình 4. Dữ liệu mẫu của bảng *Lecturers*

codeSubject	nameSubject	number of Slots
MAE101	Mathematics for Engineering	30
PFP203	Fundermental Programming with Python	30
CEA201	Computer Organization and Architecture	30
CSI104	Introduction to computing	40
PRF107	Fundermental Programming with C	30
MAD101	Discrete Mathematics	25
TRS6	English level 6	40
CSD203	Data structure and Algorithms with Python_Cấu trúc dữ liệu và giải thuật với Python	30
AIG201c	Artificial Intelligence_Trí tuệ nhân tạo (Coursera)	5

Hình 5. Dữ liệu mẫu của bảng *Subjects*

codeSemester	nameSemester	number of Weeks
SP22	Spring 2022	10
SU23	Summer 2023	15

Hình 6. Dữ liệu mẫu của bảng *Semesters*

codeSemester	Major	Subjects
SP22	AI	MAE101, PFP203, CEA201, CSI104
SP22	SE	MAE101, PRF107, CEA201, CSI104
SU23	SE	MAD101, TRS6, CEA201, CSD203
SU23	AI	MAD101, TRS6, AIG201c, CSD203

Hình 7. Dữ liệu mẫu của bảng *Syllabuses*

nameRoom	typeRoom	Description
012	Classroom	NULL
113	Seminar	NULL
234	Classroom	20 desks
220	Classroom	NULL
610	Classroom	NULL

Hình 8. Dữ liệu mẫu của bảng *Rooms*

- PHÂN TÍCH, XỬ LÝ DỮ LIỆU ĐẦU VÀO:

- + Tạo các lớp: *class* **Teacher**, *class* **Classes**, *class* **Room**, ... để lưu các thông tin của mỗi đối tượng (giảng viên, lớp học, môn học, ...)

- + Tạo biến Data (type: *class* **Data**):

- + Lưu các danh sách: các phòng học, các lớp học, các giảng viên, các môn học, thông tin về học kỳ đang cần xếp thời khóa biểu (tên học kỳ, số tuần học trong học kỳ, số slot học trong 1 ngày, số ngày học trong 1 tuần,...) thu được từ dữ liệu đầu vào.

- + Data.rooms, Data.classes, Data.teachers, ...

- + Từ biến **Data**, truy xuất thông tin để phân công số slot giảng dạy cho từng giảng viên ứng với mỗi môn, số slot học ở mỗi môn của từng lớp:

- + Lưu thông tin mỗi lớp cần học những môn gì và số slot học trong 1 tuần là bao nhiêu trong *dictionary*: **classSubjectsNumSlotWeek** = {class_1: {subject_1 : numSlotWeek, subject_2: numSlotWeek,...}, ... }

- + Kết quả phân công được lưu trong *list*: **listAssign** = [(class, subject, teacher, number_of_slots_a_week) , (), ...]

- + Tạo các lớp: **Individual**, **Population** để lưu các thông tin của mỗi đối tượng (cá thể, quần thể, ...)

- + Với mỗi cá thể, lưu thông tin chi tiết về thời gian, địa điểm học của từng lớp với từng giảng viên trong *list*: **Individual.chromosomes** = [(class_1, subject_1, day, slot, room, label: 1/..), (class_1, subject_1, day, slot, room, label: 2/..), (), ...].

Ví dụ: label 2/3 cho biết tuple này biểu diễn slot học thứ 2/3 của môn học đó trong tuần.

+ Với mỗi quần thể, lưu danh sách các cá thể trong *list*:
Population.individuals = [Individual_1, Individual_2, ...]

2. Sơ đồ giải thuật và hiện thực

- Load dữ liệu và kiểm tra. Nếu dữ liệu không hợp lệ, báo lỗi yêu cầu người dùng nhập lại file và thoát chương trình,
- Thêm dữ liệu vào các *object* với *class* tương ứng.
- Lưu các *object* này vào các *list* tương ứng trong *Class Data*.
- Lấy thông tin về học kỳ cần sắp xếp thời khóa biểu: Tên học kỳ, số slot/ngày , số ngày/ tuần,... Kiểm tra thông tin hợp lệ. Nếu không hợp lệ, yêu cầu người dùng nhập lại với học kỳ đã có sẵn trong cơ sở dữ liệu.
- Lưu các thông tin vào *Class Data*.
- Tiến hành giao việc cho giảng viên (giao cho mỗi giảng viên phụ trách dạy các môn nào của lớp nào). Sử dụng *giải thuật chia đủ theo nhu cầu – xem bên dưới*), lưu vào *dictionary listAssign*.
- Tiến hành thực hiện *giải thuật di truyền* để tìm ra cá thể tốt hơn sau mỗi thế hệ tiến hóa. (*xem bên dưới*)
- In ra thời khóa biểu cụ thể cho từng lớp, từng giảng viên dựa trên cá thể tối ưu nhất sau quá trình tiến hóa.

A. GIẢI THUẬT CHIA ĐỦ THEO NHU CẦU

Giả sử cùng một môn học: CSD203, có 3 giảng viên: X, Y, Z có thể dạy; có 6 lớp cần học môn này mỗi tuần.

Các điều kiện ràng buộc như sau:

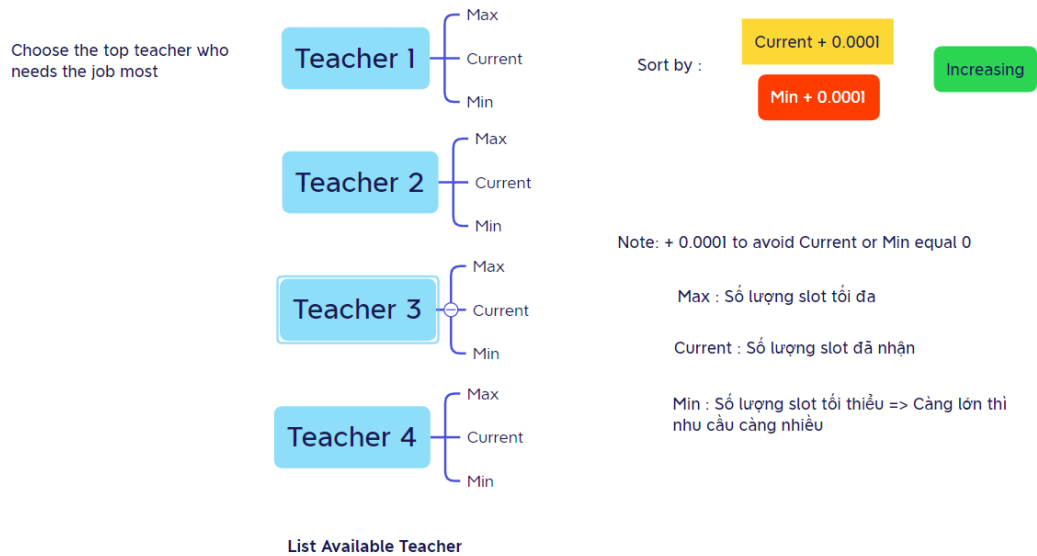
- + Môn học này học: 3 slot/tuần với mỗi lớp. Như vậy có tổng cộng 18 slot của môn học này cần chia cho các giảng viên.
- + Giảng viên X có số slot yêu cầu dạy tối thiểu là: 5, tối đa là 20 (tính trên tổng các môn dạy)
- + Giảng viên Y có số slot tối thiểu là 0, tối đa là 30.
- + Giảng viên Z có số slot tối thiểu là 10, tối đa là 15.

Yêu cầu đặt ra: Ta cần xếp số slot dạy cho các giảng viên sao cho thỏa mãn với nhu cầu của từng đối tượng – song vẫn đảm bảo sự cân đối hài hòa giữa các giảng viên (tránh trường hợp một giảng viên nhận quá nhiều slot tuy đủ nhu cầu của họ nhưng những giảng viên còn lại cũng có nhu cầu lại không được nhận slot nào).

Nhận xét:

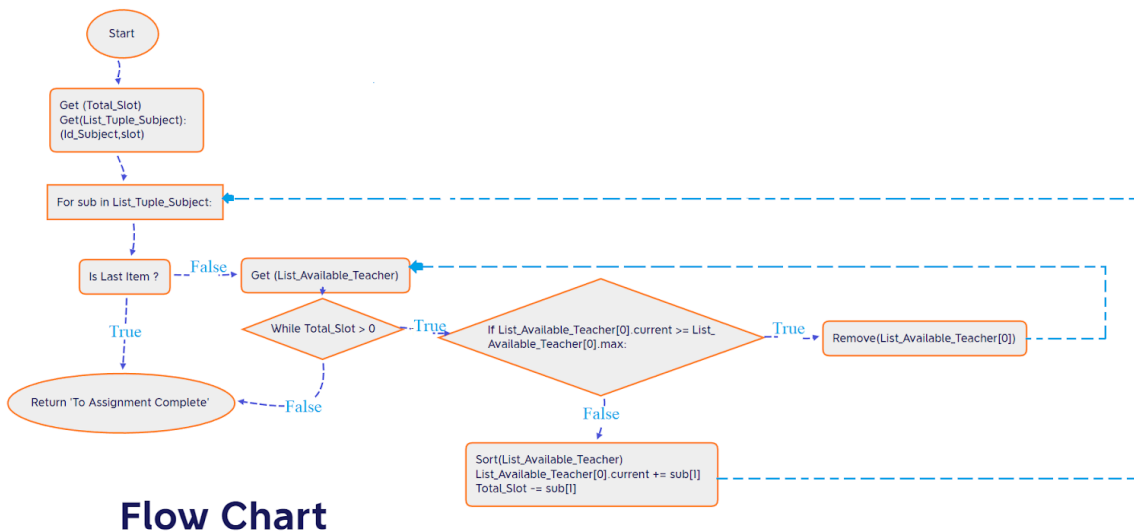
- + Giảng viên Z có số slot tối thiểu yêu cầu lớn nhất, nghĩa là nhu cầu cao nhất. Khi xếp các slot dạy cần ưu tiên xếp cho giảng viên Z trước, rồi tới giảng viên X, ...
- + Tuy nhiên, sau khi chia một số slot cho giảng viên Z, ta thấy giảng viên Z đã thỏa mãn một phần “*nhu cầu*”– trong khi giảng viên X vẫn chưa thỏa mãn phần nào (0 slot **hiện tại** / 5 slot **tối thiểu**). Do đó, lần chia tiếp theo ta cần ưu tiên xếp cho giảng viên X.

- Như vậy, ta sẽ dùng tỉ lệ: **số slot đã nhận hiện tại / số slot tối thiểu** như là thước đo sự thỏa mãn nhu cầu. Giảng viên nào có tỉ lệ này càng thấp cần nên được ưu tiên xếp slot trước.



Hình 9. Sơ đồ giảng viên

Ta có sơ đồ thuật toán như sau:



Hình 10. Sơ đồ giải thuật Chia đủ theo nhu cầu

Hiện thực giải thuật bằng Python:

```
def makeAssignList(Data):
    listAssign = []
    for subject, b in Data.subjectAndClasses.items():
        if b['num'] == 0:
            continue
        total = b['num'] * len(b['classes'])
        Data.subjectAndClasses[subject]['teachers'] = []
        for teacher in Data.teachers:
            for k in teacher.subjects:
                if k == subject.code:

Data.subjectAndClasses[subject]['teachers'].append(teacher)
        if Data.subjectAndClasses[subject]['teachers'] == []:
            print('Not enough lecturer for the subject: ', subject.code)
            quit()
        sumMin = 0
        sumMinTuple = 0
        sumMax = 0
        for teacher in Data.subjectAndClasses[subject]['teachers']:
            sumMin += teacher.minSlot
            if teacher.minSlot % b['num'] == 0:
                sumMinTuple += teacher.minSlot // b['num']
            elif teacher.minSlot % b['num'] != 0:
                sumMinTuple += teacher.minSlot // b['num'] + 1
            sumMax += teacher.maxSlot
        if sumMax < total:
            print('Not enough lecturer for the subject: ', subject.code)
            quit()
        for teacher in Data.teachers:
            teacher.numSloteachSubject[subject] = 0
        teacherList = Data.subjectAndClasses[subject]['teachers'].copy()
        teacherList = sorted(teacherList, key=lambda x: x.minSlot,
reverse=True)
        while total > 0 and teacherList != []:
            teacherList = sorted(teacherList, key=lambda x: (x.currentSlot
+ 0.0001) / (x.minSlot + 0.0001), reverse=False)
            if teacherList[0].currentSlot + b['num'] <=
teacherList[0].maxSlot:
                teacherList[0].currentSlot += b['num']
                teacherList[0].numSloteachSubject[subject] += b['num']
                total -= b['num']
            else:
                teacherList.remove(teacherList[0])
        if total > 0:
            print('Not enough lecturer for the subject:', subject.code)
            quit()
        classListtoAssign =
Data.subjectAndClasses[subject]['classes'].copy()
        for teacher in Data.subjectAndClasses[subject]['teachers']:
            numOfClass = teacher.numSloteachSubject[subject] // b['num']
            for j in range(numOfClass):
```

```

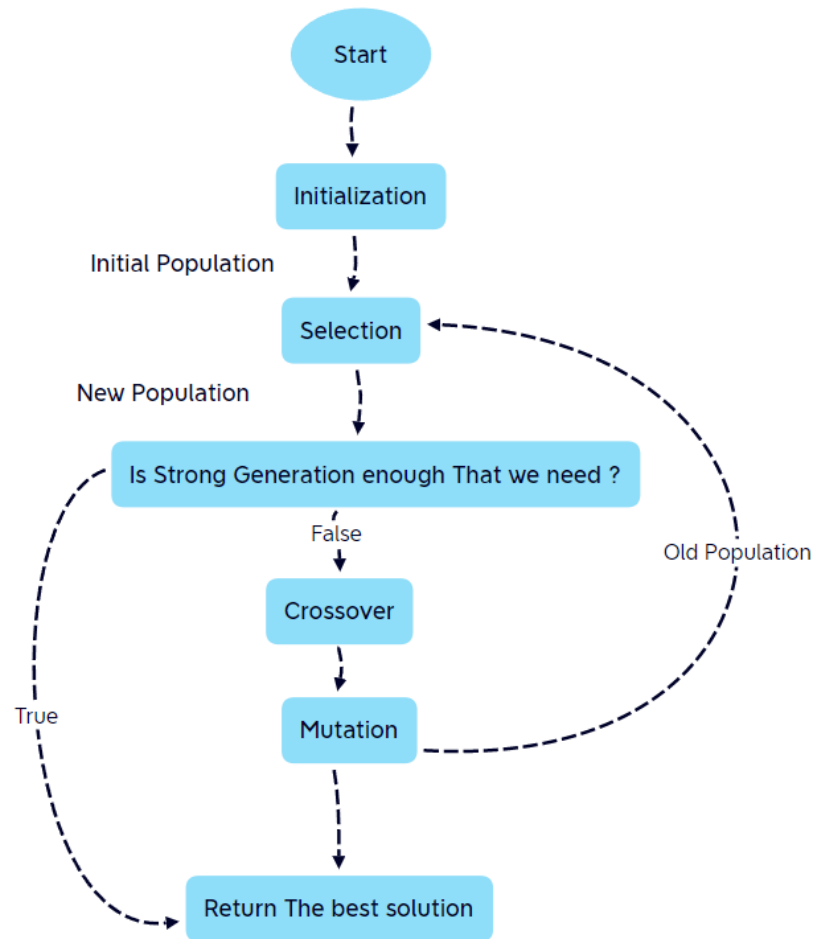
        which = random.randint(0, len(classListtoAssign) - 1)
        cl = classListtoAssign[which]
        if (cl, subject, teacher, b['num']) not in listAssign:
            listAssign.append((cl, subject, teacher, b['num']))
            classListtoAssign.remove(cl)
    if classListtoAssign != []:
        print('Not enough lecturer.')
        quit()
    return listAssign

```

B. GIẢI THUẬT DI TRUYỀN

- ***Giới thiệu chung:***

- ✚ Giải thuật di truyền dựa trên quá trình tiến hóa của sinh vật, dựa trên khái niệm cho rằng quá trình tiến hóa - chọn lọc tự nhiên là hoàn hảo nhất, tạo ra các cá thể thế hệ sau bao giờ cũng phát triển tốt hơn thế hệ trước.
- ✚ Vận dụng các nguyên lý tiến hóa tự nhiên như di truyền, đột biến, chọn lọc tự nhiên và trao đổi chéo.
- ✚ Được tạo ra để giải quyết các dạng bài toán tối ưu hóa (tìm kiếm lời giải tốt nhất trong tất cả các lời giải khả thi).



Hình 11. Lược đồ giải thuật di truyền cơ bản

- **Ứng dụng trong bài toán Sắp xếp Thời khóa biểu.**

- ✚ Ta coi mỗi một cách sắp xếp thời khóa biểu như là một cá thể (đã sắp xếp hoàn chỉnh lịch dạy và học chi tiết cho tất cả các giảng viên và các lớp – tuy nhiên thời gian và địa điểm được lựa chọn ngẫu nhiên do đó tồn tại rất nhiều sự xung đột như: cùng một ca học của một ngày, tại cùng một phòng học có nhiều hơn 1 lớp học; hoặc một lớp tại một thời điểm học 2 môn khác nhau; hoặc giảng viên phải dạy vào skip days – các ngày giảng viên không thể dạy – đặc thù riêng đối với từng giảng viên).

- ✦ Đặc trưng của mỗi cá thể như vậy là một bộ NST (một *list* gồm các bộ thông tin (*class, subject, teacher, which day, which slot, which room, label of slot*) – được gọi là một NST. Mỗi NST là một bộ thông tin về: lớp đó học môn gì, với giảng viên nào, ở phòng học nào, vào ngày nào, ca học nào và số slot hiện tại của môn đó trong tuần.
- ✦ Ta khởi tạo nhiều cá thể như vậy, sau quá trình di truyền và tiến hóa ta sẽ chọn được cá thể tối ưu nhất, chính là bộ thời khóa biểu tốt nhất và ít có xung đột nhất.

Giải thuật di truyền bao gồm 3 bước chính:

➤ **Bước 1:** KHỞI TẠO QUẦN THỂ.

- ❖ Từ một bộ gen (*list: listAssign*), tạo ra các bộ NST ngẫu nhiên (*list: chromosomes*). Mỗi bộ NST này tạo thành một cá thể đơn lập (*class: Individual*)

Chú thích:

- **listAssign** = [(lớp, môn học, giảng viên, số slot 1 tuần), (...), ...]
- **chromosomes** = [(class_1, subject_1, teacher, day, slot, room, label: 1/..), (class_1, subject_1, teacher, day, slot, room, label: 2/..), ..., ...]
- ❖ Khởi tạo một quần thể với 9 cá thể (*class: Population*)

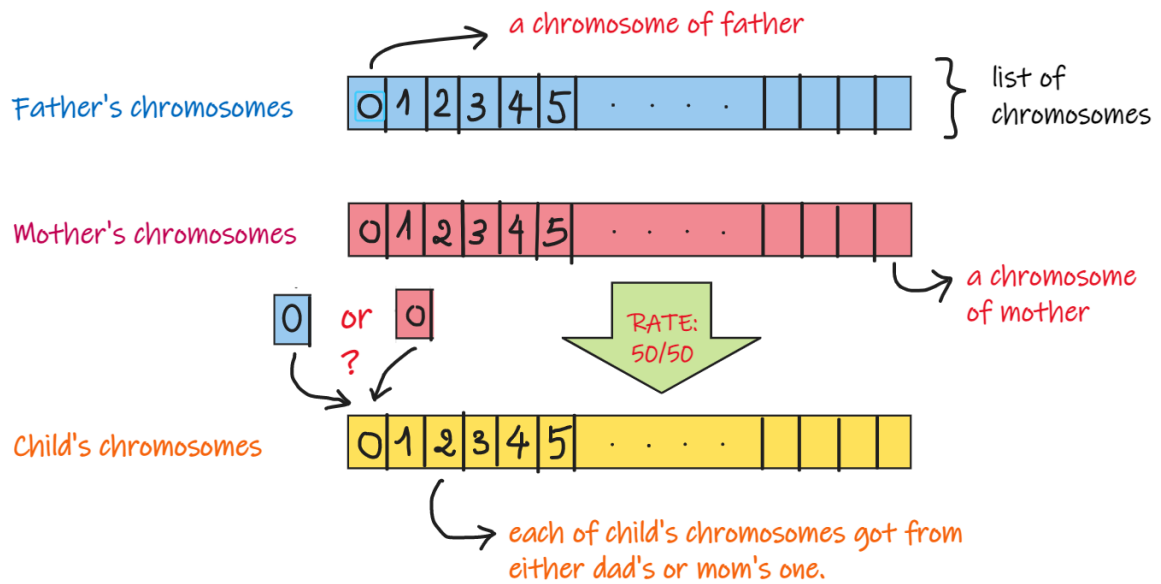
➤ **Bước 2:** SỬ DỤNG CÁC PHÉP TOÁN DI TRUYỀN.

❖ *Thay thế quần thể ban đầu bằng quần thể lai tạo:*

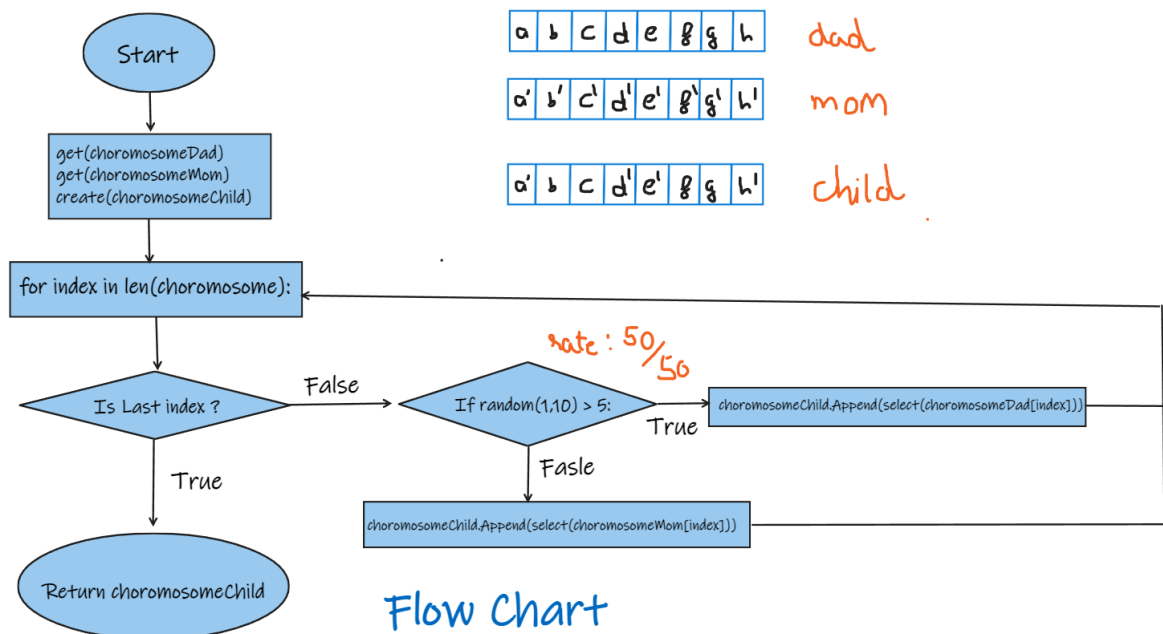
1. Giữ lại một số cá thể tối ưu nhất của quần thể ban đầu.
2. Mỗi cá thể còn lại được sinh ra bằng phép lai giữa cá thể cha và cá thể mẹ, trong đó:
 - Chọn ngẫu nhiên một nửa số cá thể trong quần thể ban đầu, chọn ra cá thể tốt nhất – gọi là *cá thể cha*.
 - Chọn ngẫu nhiên một nửa số cá thể trong quần thể ban đầu, chọn ra cá thể tốt nhất - gọi là *cá thể mẹ*.

✓ *Giải thuật Lai giữa 2 cá thể như sau:*

Từ bộ NST của cá thể cha và mẹ, mỗi NST của cá thể con sẽ được chọn ngẫu nhiên: hoặc từ NST tương ứng đó của cá thể cha, hoặc từ NST tương ứng của cá thể mẹ.



Hình 12. Mô hình Lai giữa 2 cá thể



Hình 13. Sơ đồ giải thuật Lai giữa 2 cá thể

✓ Hiện thực giải thuật Lai giữa 2 cá thể bằng Python:

```
def crossover(individual_1, individual_2):
    fatherChromosomes = individual_1.chromosomes
    motherChromosomes = individual_2.chromosomes
    childChromosomes = []
    for k in range(len(individual_1.chromosomes)):
        which = random.randint(1, 10)
        if which > 5:
            childChromosomes.append(fatherChromosomes[k])
        else:
            childChromosomes.append(motherChromosomes[k])
    individual_child = Individual(childChromosomes)
    return individual_child
```

Hiện thực giải thuật lai tạo quần thể bằng Python:

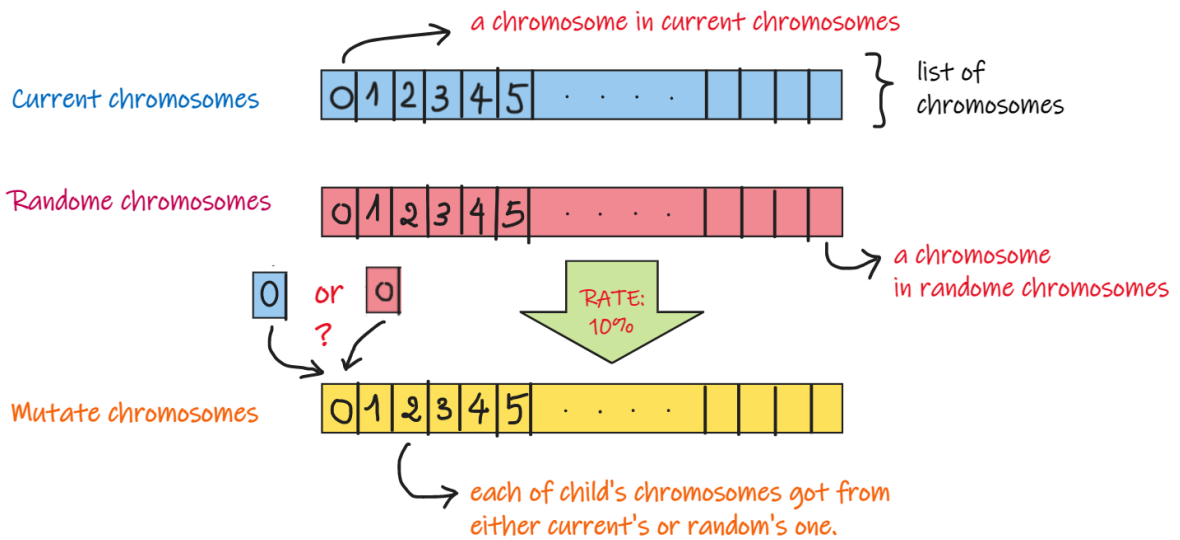
```
def makeCrossoverPop(population):
    crossoverPop = Population(Data, 0)
    population.individuals = sorted(population.individuals, key=lambda
x:x.fitness(), reverse=True)
    for k in range(NUMBER_POPULATION_SELECTION):
        crossoverPop.individuals.append(population.individuals[k])
    for k in range(NUMBER_POPULATION_SELECTION, SIZE_POPULATION, 1):
        individual_1 = chooseTopFromHalfPopulation(population)
        individual_2 = chooseTopFromHalfPopulation(population)
        individual_child = crossover(individual_1, individual_2)
        crossoverPop.individuals.append(individual_child)
    return crossoverPop
```

❖ Đột biến quần thể lai tạo:

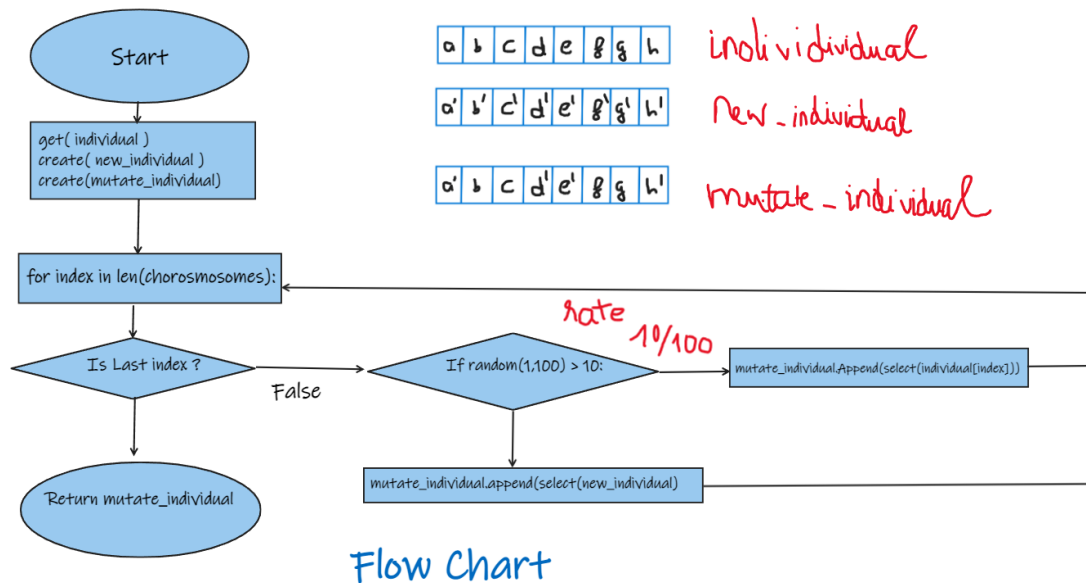
1. Giữ lại một số cá thể tối ưu nhất.
2. Mỗi cá thể còn lại được sinh ra bằng phép đột biến chính cá thể đó.

✓ Giải thuật Đột biến cá thể được thực hiện như sau:

Từ bộ NST của cá thể đó, mỗi NST sẽ được giữ lại hoặc bị thay thế bằng NST tương ứng của một cá thể mới ngẫu nhiên – dựa trên tỉ lệ đột biến.



Hình 14. Mô hình Đột biến cá thể



Hình 15. Sơ đồ giải thuật Đột biến cá thể

✓ Hiện thực giải thuật Đột biến cá thể bằng Python:

```
def mutateIndividual(currentIndividual):
    currentChromosomes = currentIndividual.chromosomes
    randomChromosomes = makeChromosomes(Data)
    mutateChromosomes = []
    for k in range(len(currentChromosomes)):
        which = random.randint(0,100)
        if which <= RATE_INDIVIDUAL_MUTATE:
            mutateChromosomes.append(randomChromosomes[k])
        else:
            mutateChromosomes.append(currentChromosomes[k])
    mutate_individual = Individual(mutateChromosomes)
    return mutate_individual
```

Hiện thực giải thuật đột biến quần thể bằng Python:

```
def mutatePopulation(population):
    mutate_population = Population(Data,0)
    population.individuals = sorted(population.individuals, key=lambda
x:x.fitness(), reverse=True)
    for k in range(NUMBER_POPULATION_SELECTION):
        mutate_population.individuals.append(population.individuals[k])
    for k in range(NUMBER_POPULATION_SELECTION, SIZE_POPULATION):
        currentIndividual = population.individuals[k]
        mutate_individual = mutateIndividual(currentIndividual)
        mutate_population.individuals.append(mutate_individual)
    return mutate_population
```

➤ **Bước 3:** ĐÁNH GIÁ ĐỘ THÍCH NGHI, CHỌN RA CÁ THỂ TỐI ƯU NHẤT.

- ❖ Kiểm tra độ thích nghi của các cá thể trong quần thể. Chọn ra cá thể tốt nhất.
- ❖ Thực hiện lặp lại **Bước 2** cho đến khi cá thể tốt nhất trong quần thể đạt tiêu chuẩn theo yêu cầu (không chứa xung đột hoặc là cá thể tối ưu nhất sau số thế hệ tiến hóa quy ước trước).

Hiện thực giải thuật đánh giá độ thích nghi và tiến hóa bằng Python:

```
class Individual:
    def __init__(self, chromosomes):
        self.chromosomes = chromosomes
    def fitness(self):
        score = 0
        check_conflicts_Lecturer = []
        check_conflicts_Location = []
        check_conflicts_Class = []
        for k in self.chromosomes:
            if (k[3],k[4],k[5]) in check_conflicts_Location:
                score -= 1
            else:
                check_conflicts_Location.append((k[3], k[4], k[5]))
            if (k[0],k[3],k[4]) in check_conflicts_Class:
                score -= 1
            else:
                check_conflicts_Class.append((k[0], k[3], k[4]))
            if (k[2],k[3],k[4]) in check_conflicts_Lecturer:
                score -= 1
            else:
                check_conflicts_Lecturer.append((k[2],k[3],k[4]))
            if k[3] in k[2].skipDays:
                score -= 1
        return score

while population.individuals[0].fitness() < 0:
    number_of_generation += 1
    print('Thế hệ thứ: ', number_of_generation)
    print('fitness của cá thể mạnh nhất đến nay là: ',
population.individuals[0].fitness())
    crossOverPop = makeCrossoverPop(population)
    population = mutatePopulation(crossOverPop)
    population.individuals =
sorted(population.individuals, key=lambda x: x.fitness(),
reverse=True)
```

3. Kết quả và thảo luận

- Với dữ liệu đầu vào chúng em cung cấp, sau khoảng 70 - 500 thế hệ tiến hóa (thời gian trung bình: 1,4934598s), chương trình đã thu được cá thể đạt tiêu chuẩn, thỏa mãn các ràng buộc cơ bản theo yêu cầu.
 - Nếu dữ liệu đầu vào quá khắt khe, chương trình có thể chạy tới 10,000 lần nhưng vẫn còn xung đột.
- *Thảo luận:* Nếu số phòng học hoặc số slot học trong ngày hoặc số ngày học trong tuần quá ít thì sẽ khó thỏa mãn tất cả các ràng buộc, dẫn tới vòng lặp vô hạn (bài toán không có lời giải tốt nhất). Mặt khác, vì là phân bố ngẫu nhiên nên vẫn có sự “*may rủi*” trong các lần chạy, dẫn tới sự xê dịch thời gian chạy chương trình; tuy nhiên khoảng thời gian này là không đáng kể và thuật toán di truyền – tiến hóa đảm bảo chất lượng cá thể luôn đi lên (hoặc ít nhất là không giảm) sau mỗi thế hệ.

TỔNG KẾT

- ❖ Bài toán sơ bộ đã giải quyết được các vấn đề cơ bản của việc sắp xếp thời khóa biểu sao cho hợp lý nhất và luôn tuân thủ các điều kiện cơ bản của giảng viên và sinh viên. Hiện thị đầy đủ các chức năng như: in thời khóa biểu cho từng lớp và từng giảng viên; hiện thị được bảng kế hoạch công việc cho cả tuần học; ...

Một số tính năng mong muốn cải thiện :

- ✓ Được xây dựng trên một ứng dụng hoàn chỉnh (sử dụng thư viện **Tinker**) và có giao diện người dùng đầy đủ, thuận tiện cho việc xử lý dữ liệu.
- ✓ Load được dữ liệu từ database (sử dụng thư viện **sqlite3**)
- ✓ Ứng dụng mã hóa dữ liệu đầu vào để tiết kiệm bộ nhớ, tối ưu hóa giải thuật.
- ✓ Ứng dụng thư viện có sẵn cho giải thuật di truyền (sử dụng thư viện **PyGAD**).
- ❖ Sau bài tập này, chúng em đã được tìm hiểu cách quản lý cơ sở dữ liệu sao cho hiệu quả nhất; nghiên cứu và phân tích các giải thuật chưa từng được tiếp cận và ứng dụng những kiến thức đó vào bài tập của mình. Ngoài ra, chúng em cũng có cơ hội được mài giũa kỹ năng làm việc nhóm và sử dụng các phần mềm tiện ích phục vụ cho làm việc cộng tác như: Google Docs, GitHub, Google Meeting, ...

TÀI LIỆU THAM KHẢO

Cần Kim Tùng, Đào Trọng Huân, và cộng sự (2013). *Giải thuật di truyền với bài toán người du lịch*. Trường Đại Học Bách Khoa Hà Nội, Hà Nội.

Clinton Sheppard (2016). *Genetic Algorithms with Python*.

Zane Academy [Prototype Project]. (2017, June 14). *Genetic Algorithms (01) Python Prototype Project* [Video]. YouTube:

https://www.youtube.com/watch?v=8NrNX_jCkfw.

Zane Academy [Prototype Project]. (2019, July 19). *Genetic Algorithms (02) + Class Scheduling (01) – Python Prototype Project* [Video]. YouTube:

https://www.youtube.com/watch?v=8NrNX_jCkfw.

Nero Phung (2020, May 27). *Genetic Algorithm – Giải thuật di truyền*. Khai thác từ: <https://nerophung.github.io/2020/05/28/genetic-algorithm>

TỰ ĐÁNH GIÁ

Nội dung	Điểm	Ghi chú
Giới thiệu về bài toán (0.25 đ)	0.25 đ	
Mô tả cấu trúc dữ liệu (1.25 đ)	1.25 đ	
Sơ đồ giải thuật (1.5 đ)	1.5 đ	
Hiện thực (5 đ)	5 đ	
Kết quả và thảo luận (0.5 đ)	0.4 đ	Trong một số trường hợp, chưa thu được hiệu quả tối ưu nhất.
Điểm nhóm (0.75 đ)	0.75 đ	
Các điều rút ra cho bản thân (0.25 đ)	0.15 đ	Chúng em nghĩ chúng em cần rút ra được nhiều bài học hơn.
Báo cáo đúng theo mẫu (0.5 đ)	0.5 đ	
Tổng điểm	9.8 đ	