**Báo cáo-nhóm 9**

**Bài tập về nhà**

****

* Để thuận tiện cho việc kiểm tra, đánh giá code của các nhóm cũng như để so sánh và hiểu rõ hơn về cách thuật toán **Tham lam** hoạt động. Chúng em đã viết một vài đoạn code như sau:

+ Các hàm giải quyết bài toán **Knapsack Problem** sử dụng kỹ thuật **Greedy Appoach**.

+ Tạo bộ testcase đơn giản bằng thư viện random.

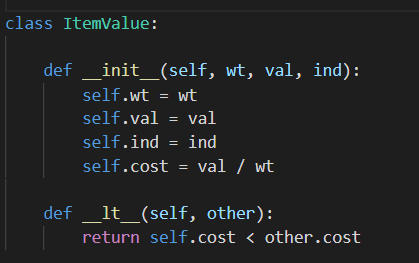
+ Hàm kiểm tra và chấm điểm code các nhóm submit dựa trên testcase.

* + - 1. Testcase nhỏ để kiểm tra hoạt động chương trình.
      2. Testcase lớn để kiểm tra hiệu năng chương trình.

1. Giải quyết Knapsack Problem bằng Greedy Appoach

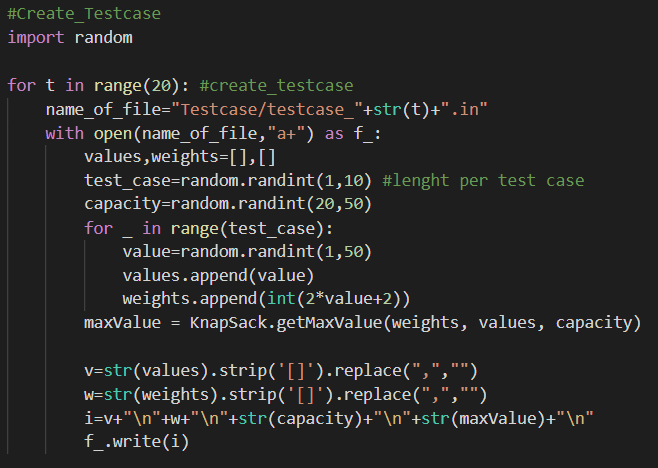
Theo như ý tưởng cũng như giải thuật đã trình bày, giới thiệu trên buổi báo cáo trực tiếp trên lớp học. Bài toán giải quyết bằng kỹ thuật tham lam và tiếp cận theo tỉ số Trị giá / Cân nặng từ lớn tới nhỏ.

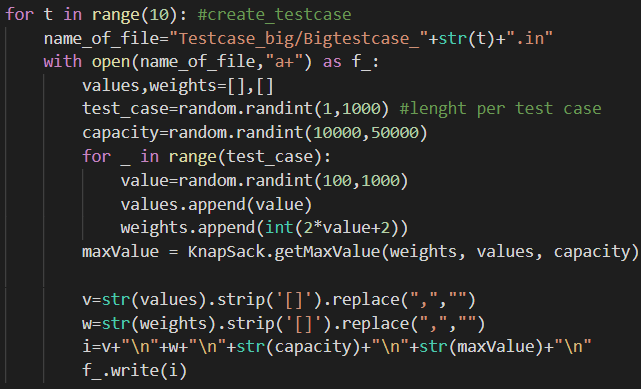
* + Một class nhỏ để quản lý các thuộc tính về trị giá, cân nặng, tỉ số.



* + Một class Knapsack và hàm giải quyết kèm theo.



1. Tạo Testcase
   * Ở đây em tạo bộ testcase đơn giản bằng hàm random và không cần sử dụng thư viện unitest.
   * Sử dụng hàm random để tạo các giá trị như trị giá, số lượng mặt hàng trong mỗi testcase,…
   * Bộ test case nhỏ gồm có 20 test case có tối đa 10 cặp value/weight được lưu thành 20 file .txt
   * Bộ test case lớn gồm 10 test case có tối đa 1000 cặp value.weight được lưu thành 10 file .txt

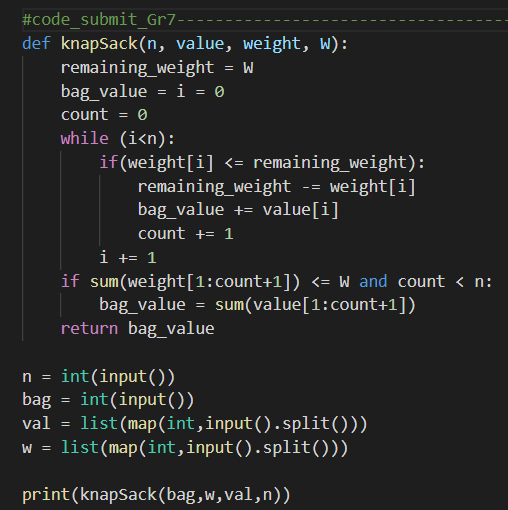


* + Để đơn giản hóa và trực quan dữ liệu giữa trị giá và cân nặng cho phù hợp, em đã chọn hàm số bậc nhất y=2.x + 2 để mô tả cho quan hệ này. Điều này có thể gây ra sự tương đồng giữa các cặp trị giá – cân nặng, nhưng nó lại dể mô tả, trong lúc kiểm tra code.

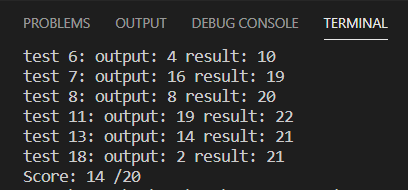
1. Kiểm tra code các nhóm submit
   * Hàm kiểm tra kết quả giữa code submit và testcase



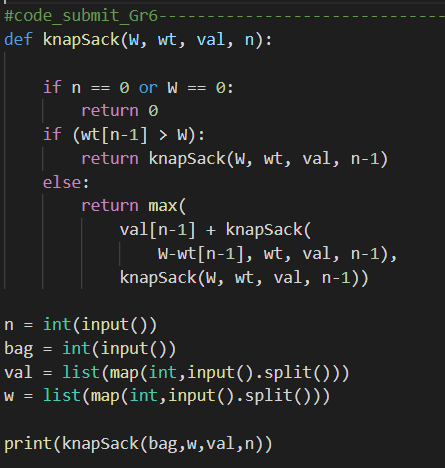
* + Sau đây là demo của các nhóm đã nộp bài.
* Nhóm 7:



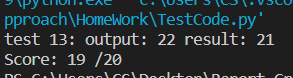
* + Nhận xét: Code dựa trên ý tưởng cũng như cách tiếp cận theo hướng Tham lam, cộng giá trị của bag theo giới giạn của W (maxWeight) dựa trên tỉ số giữa trị giá và cân nặng
  + Kết quả:



* + - Kết quả tương đối khả quan đối với một kỹ thuật tối ưu cục bộ để tìm ra tối ưu toàn cục của Tham lam.
    - Nguyên nhân và cách khắc phục các test sai sẽ được nhóm giải đáp vào đầu buổi học tiếp theo.
* Nhóm 6:



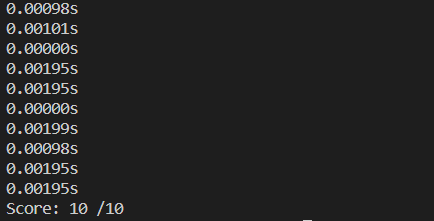
* + Nhận xét: Code nhóm 6 khá ngắn gọn dể hiểu, nhưng đặt biệt là nhóm không tiếp cận Tham lam theo hướng trị số chia cho cân nặng mà tiếp cận theo hướng tìm giá trị lớn nhất giữa các nhóm phần tử bằng đệ quy.
  + Kết quả:



* + - Thật bất ngờ khi thuật toán cho ra kết quả giống với testcase đến 19/20 test và test còn lại không sai mà còn tối ưu hơn cả testcase.
    - Tuy nhiên thuật toán này sử dụng kỹ thuật đệ quy loại bỏ từng phần tử để tìm ra maxValue, điều này có thể tốn rất nhiều chi phí so với việc sử dụng Tham lam. Như đã đề cập trước đó, Tham lam rất nhanh đồng nghĩa với việc chấp nhận một vài sai lệch, thậm chí nếu bước tối ưu cục bộ không thực sự hiểu quả có thể dẫn đến một kết quả hoàn toàn sai lệch.
    - Nhóm sẽ trình bày và thảo luận về kết quả của thuật toán tham lam với kết quả của nhóm 6 trong buổi học sau.

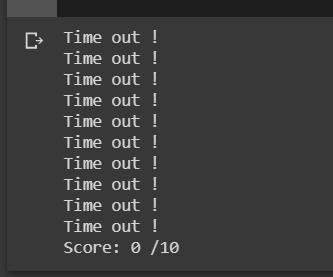
Nhận xét: với testcase nhỏ, đa phần các nhóm đều đạt được kết quả tương đối trên 50%. Do có một vài nhóm không sử dụng hướng tiếp cận tham lam mà thay vào đó là sử dụng kỹ thuật đệ quy quét toàn bộ danh sách.

* Testcase lớn (tối đa 1000 cặp value/weight)
  + Tham lam.



Thời gian thực thi rất ấn tượng !

* + Đệ quy.



Do phải sài thư viện để set thời gian time out nên phải sử dụng colab ( thư viện này không chạy được trên window). Thới gian thực thi tối đa là 60s tuy nhiên toàn bộ test case đều bị time out. Điều này cũng không quá bất ngờ khi bắt đệ quy thực hiện với một test case quá lớn như vậy.

**Tóm lại:**

* **Giải quyết bài toán bằng tham lam rất nhanh, dể cài đặt.**
* **Nếu không yêu cầu về độ chính xác tuyệt đối, ở mức tham khảo xấp xỉ kết quả thì tham lam thực sự hiệu quả.**