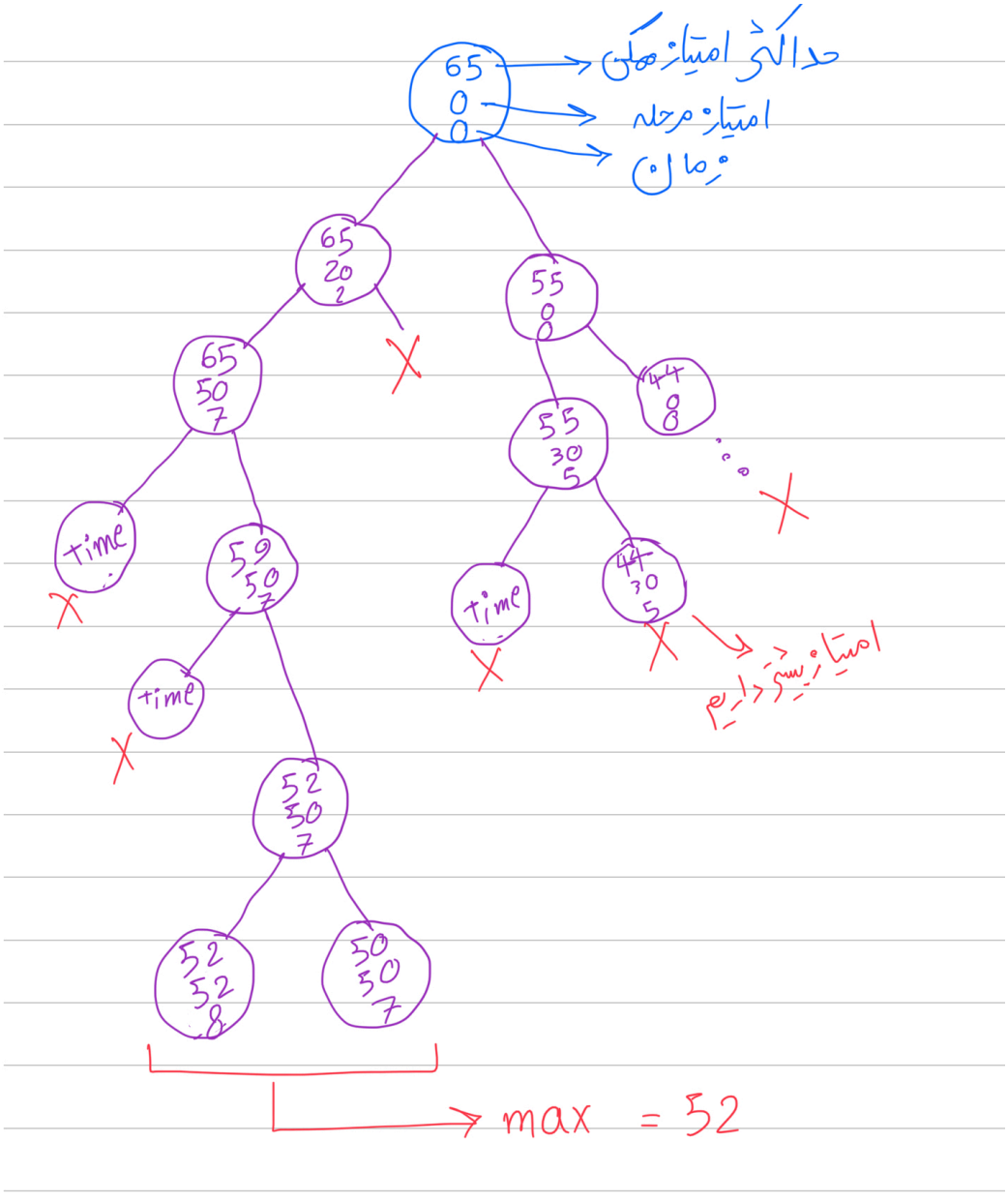
سید نامی مدرسی
۹۷۳۳۰۶۹
زمستان ۱۴۰۰

کوییز

سوال ۱)

- بخش اول : در این بخش مانند بخش بر اساس نسبت جایزه به زمان انتخاب می کنیم و هر بار چک می کنیم اگر انتخاب کنیم صندق را چه می شود و اگر نه چه و درخت را تشکیل می دهیم در نهایت یک گره می ماند که نهایی است .



- بخش دوم : این همان مسئله کوله پشتی (حالت کسری) است که با روش حریصانه قابل حل است یعنی ما بیایم و صندوق ها را بر اساس نسبت جایزه به زمان مرتب کنیم و آن صندقی که این نسبتش بیشتر از بقیه است را اول انتخاب کنیم وبعد به صندوق بعدی برویم . در نهایت با این روش می توانیم به جواب بهینه برسیم .

S_i	T_i	ارزش
20	2	10
30	5	6
35	7	5
12	4	3
2	1	2

ثانیه 10 $\rightarrow$: $20 + 30 + \underbrace{\frac{3}{7} \times 35}_{15} = 65$

سوال ۲)

برای این سوال روش حریصانه استفاده کردم .

در این روش ما دنبال حداکثر کردن سود هستیم پس در هر مرحله باید پردازه ای را انتخاب کنیم که سود بیشینه دارد.

پس الگوریتم به این شکل است که در هر مرحله ما پردازه ای که بیشترین سود را دارد انتخاب می کنیم (مثلا اول کار آن ها را مرتب می کنیم یعنی پردازه با سود بیشتر بالاتر باشد) و بعد آن ها را انتخاب می کنیم .

برای مثال داخل سوال :

سوال ۳)

اگر سکه ای به ارزش ۱ داشته باشیم به شکل حریصانه حل می کنیم به این شکل که در هر مرحله تا جایی که می شود بزرگترین سکه ای که می توان خرج کرد را خرج می کنیم .

اگر سکه به ارزش ۱ نداشته باشیم از روش برنامه نویسی پویا استفاده می کنیم :

به عنوان حافظه یک آرایه دو بعدی در نظر میگیریم که ستون آن سکه های di هستند و سطر آن هم مجموع پول آن سکه ها (سکه های d0 تا di ، i شمارنده آن ستون) پس یعنی یک خانه مثل [i][j] تعداد سکه هایی است که از d0 تا dj بوده و مجموع آن ها i باشد .

حال در هر مرحله یا سکه بعدی را انتخاب می کنیم یا رد می شویم یعنی سکه را انتخاب نمی کنیم.

پس رابطه بین خانه ها :

$$[I][j] = \min([I-di][j] + 1 , [j][i-1])$$

رابطه به این معنی است که یا ما یک سکه مثلا di را بر می داریم یا بر نمی داریم .

اگر برنداریم تعداد سکه ها و مبلغی که باید پرداخت کنیم فرقی نمی کند .

اگر برداریم تعداد سکه ها یک واحد زیاد می شود و هزینه هم به اندازه ارزش آن کم می شود .

در نهایت در خانه اخر یعنی m,n جواب نهایی را پیدا می کنیم .

سوال ۴)

- تقسیم و غلبه : این روش روش پایه ای برای تقسیم کار هاست به این معنی که اگر بتوانیم یک سوال را به قسمت هایی بشکنیم و آن ها را حل کنیم و در نهایت بتوانیم آن ها را ترکیب کنیم از این روش استفاده می کنیم . یعنی یک مسئله با اندازه n را به مسئله های با اندازه مثلا n/k تبدیل می کنیم و آن ها را حل می کنیم .

- برنامه نویسی پویا : این روش بهبودی برای روش قبلی است با این دید که در روش قبلی ممکن است ما یک حالت را چندین بار حل کنیم که این کار باعث می شود بهینه عمل نکنیم پس بهتر است از یک حافظه استفاده کنیم و مواردی که محاسبه کردیم را در یک حافظه نگه می داریم تا بعدا بتوانیم از مقادیر آن استفاده کنیم یعنی در اینجا ما از مسئله با اندازه های کوچک به سمت مسئله با اندازه n حرکت می کنیم .

- مقایسه این دو روش : در هر دو روش ما به شکل بازگشتی مسئله را حل می کنیم منتها در روش تقسیم و غلبه ما ممکن است مسئله برای یم مقدار را چندین بار محاسبه کنیم ولی در روش برنامه نویسی پویا ما آن را در یک حافظه قرار می دهیم و اگر بعدا نیاز شد از آن استفاده می کنیم و دلیل این موضوع این است که در روش برنامه نویسی پویا زیر مسئله ها اشتراک دارند .

- عقبگرد : این روش متفاوت از دو روش قبلی است زیرا در اینجا ما به دنبال تقسیم مسئله نیستیم . این روش برای زمانی است که ما می خواهیم همه حالت ها را بگردیم یعنی فضای حالت را چک کنیم تا یک سری جواب برای متغیر ها پیدا کنیم (مثل مسئله ارضای محدودیت csp یعنی دنبال جوابی هستیم که یک سری شرط را برقرار کند) .