طراحی الگوریتم

جزوه ششم - NP and NP-Complete

در این بخش به بررسی کلاس‌های مختلف پیچیدگی مسائل می‌پردازیم. واضح است که مسائل P زیرمجموعه‌ی مسائل NP هستند و یا به عبارتی هر مساله‌ی عضو P در دسته‌ی NP نیز قرار دارد ولی برابر یا نابرابر بودن P و NP مساله‌ای است که هنوز ثابت نشده و اثبات آن تحولی در دنیای کامپیوتر خواهد بود. در ادامه‌ی این مبحث به انجام تقلیل، کاهش یا تحویل (reduction) مسائل می‌پردازیم؛ هدف از تقلیل این است که نشان دهیم یک مساله حداقل به سختی مساله‌ی دیگر است. اگر مساله‌ی X را به مساله‌ی Y کاهش دهیم، حل مساله‌ی Y، حل X را نتیجه می‌دهد. این موضوع که Y حداقل به اندازه‌ی X سخت است را به صورت $X \leq Y$ نمایش می‌دهیم. مسائل $NP - Hard$ مسائلی هستند که اگر بتوانیم آنها را در زمان چندجمله‌ای حل کنیم، می‌توانیم تمام مسائل دسته‌ی NP را در زمان چندجمله‌ای حل کنیم.

۱. 3-SAT (3-Conjunctive Normal Form or 3-CNF)

ورودی این مساله، n متغیر از جنس Boolean است و همچنین m عدد عبارت (clause) داریم که و هرکدام به صورت AND سه متغیر و یا نقیض آنها نوشته می‌شود؛ برای مثال $x_1 \vee \overline{x_2} \vee x_4$ یک نمونه عبارت است که شامل اپراتورهای $\vee$ و NOT است. ثابت کنید تعیین این موضوع که آیا می‌توان مقادیر ۰ یا ۱ را به گونه‌ای به ورودی‌ها اختصاص داد که خروجی برابر با true شود، مساله‌ای است که در دسته‌ی NP-complete قرار می‌گیرد. مثالی از 3-SAT به صورت $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$ است.

پاسخ :

برای اثبات NP-complete بودن این مساله دو گام زیر را انجام می‌دهیم:

۱) اثبات NP بودن مساله‌ی 3-SAT: گواهی داده شده شامل تخصیص مقادیر ۰ و ۱ به ورودی‌های مساله است و بدیهی است بررسی صحت آن در زمان چندجمله‌ای امکان‌پذیر است و درنتیجه این مساله عضو دسته‌ی NP است.

۲) اثبات NP-Hard بودن مساله‌ی 3-SAT: برای اثبات این موضوع، مساله‌ی Circuit-SAT که می‌دانیم یک مساله‌ی NP-Complete است را به آن کاهش می‌دهیم. برای هر مدار دودویی چهار گام زیر را انجام می‌دهیم:

- شکستن هر گیت به گیت‌هایی که شامل دو ورودی است
- ذخیره کردن نتایج میانی در متغیرها

- نوشتن معادله‌ی دودویی متناظر با مدار دودویی: این مرحله به این صورت انجام می‌شود که متغیری که به خروجی یک گیت اختصاص داده شده درست خواهد بود اگروتنها اگر عملیات انجام شده بین دو متغیر متناظر با ورودی آن گیت درست باشد؛ برای مثال در صورتی که گیت OR داشته باشیم و متغیرهای ورودی آن برابر با x_1 و نقیض x_2 و متغیر خروجی متناظر با آن برابر با x_3 باشد، معادله‌ی آن را به صورت $x_3 \leftrightarrow x_1 \vee \overline{x}_2$ نوشته می‌شود.

- فرمول نهایی زمانی satisfiable است که تمام معادلات میانی satisfy شده باشند؛ بنابراین می‌توانیم فرمول نهایی را با قرار دادن AND بین معادلات میانی مرحله‌ی قبل بدست آوریم؛ برای مثال درصورتی که معادلات میانی به صورت $(x_3 \leftrightarrow x_1 \vee \overline{x}_2)$ ، $(x_2 \leftrightarrow \overline{x}_5 \vee x_4)$ و $(x_4 \leftrightarrow x_6 \vee x_7)$ باشند، معادله‌ی نهایی برابر با $(x_3 \leftrightarrow x_1 \vee \overline{x}_2) \wedge (x_2 \leftrightarrow \overline{x}_5 \vee x_4) \wedge (x_4 \leftrightarrow x_6 \vee x_7)$ خواهد شد.

پس از انجام مراحل ذکر شده، ردیف‌هایی از جدول درستی معادلات میانی که نتیجه‌ی آن برابر با ۰ شده را در نظر می‌گیریم و clause مربوط به آنها را نوشته و با یکدیگر AND می‌کنیم؛ در نهایت نقیض عبارت را با استفاده از قانون دمورگان بدست می‌آوریم تا ردیف‌هایی که نتیجه‌ی آنها برابر با true است در نظر گرفته شود و اپراتور بین clause های میانی برابر با OR شود. در انتهای این مرحله clause هایی بدست می‌آید که لزوما شامل سه متغیر نیست.

- درصورتی که clause دارای دو متغیر بود، متغیر جدیدی مانند p اضافه می‌کنیم به گونه‌ای که true یا false بودن آن تاثیری در نتیجه‌ی clause نداشته باشد. برای این کار یک clause را تبدیل به AND دو clause می‌کنیم و در یکی از آنها p و در دیگری نقیض p را قرار می‌دهیم. برای مثال:

$$(l_1 \vee l_2) \rightarrow (l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \overline{p})$$

- درصورتی که clause دارای یک متغیر بود، دو متغیر جدید مانند p و q اضافه می‌کنیم به طوری که true یا false بودن آنها تاثیری در نتیجه‌ی clause نداشته باشد. برای این کار یک clause را تبدیل به AND سه clause می‌کنیم. برای مثال:

$$l \rightarrow (l \vee p \vee q) \wedge (l \vee p \vee \overline{q}) \wedge (l \vee \overline{p} \vee q) \wedge (l \vee \overline{p} \vee \overline{q})$$

با تبدیل ورودی Circuit-SAT به ورودی‌های 3-SAT و کاهش آن اثبات کردیم که مساله‌ی 3-SAT، NP-Hard است و از آنجایی که NP است، NP-Complete خواهد بود.

برای اثبات NP-Hard بودن یک مساله، مساله‌ی دیگری که NP-Complete بودن آن اثبات شده است را به مساله‌ی مورد نظر کاهش می‌دهیم. نکته‌ای که در این مساله مطرح است، جهت کاهش است و با کاهش یک مساله‌ی NP-Complete به مساله‌ی مورد نظر در زمان چندجمله‌ای در واقع نشان می‌دهیم که تمام مسائل NP را می‌توان در زمان چندجمله‌ای به مساله‌ی مورد نظر کاهش داد و درنتیجه NP-Hard است.

✓ مساله‌ای NP-Complete است که هم NP و هم NP-Hard باشد؛ برای اثبات NP-Complete بودن آن، باید هم NP و هم NP-Hard بودن مساله اثبات شود.

۲. CLIQUE Problem

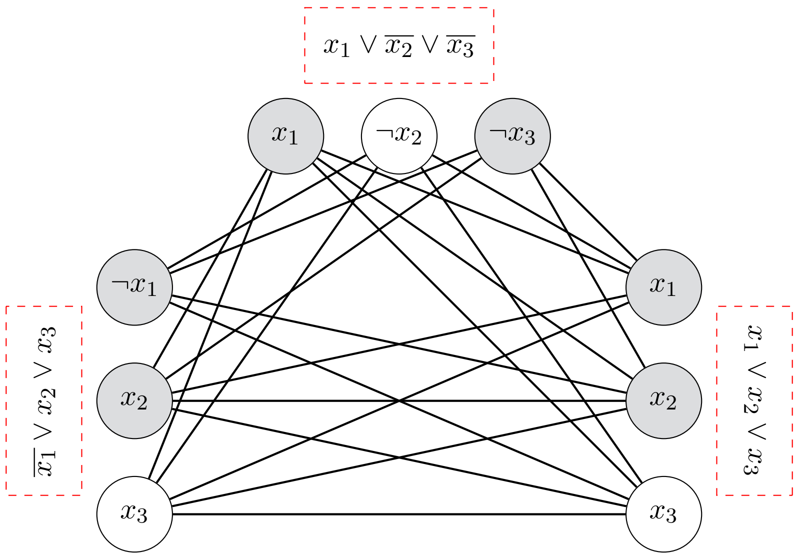
گراف بدون جهت G و مقدار k را در نظر بگیرید؛ مسالهی تصمیم گیری CLIQUE بیان می‌کند که آیا می‌توان k راس را در این گراف پیدا کرد به طوری که همگی همسایه‌ی یکدیگر باشند و یا به عبارتی هر راس در این k راس به $k-1$ راس دیگر یال داشته باشند. ثابت کنید که این مساله در دسته مسائل NP-Complete قرار دارد.

پاسخ :

برای اثبات NP-complete بودن این مساله دو گام زیر را انجام می‌دهیم:

۱) اثبات NP بودن مساله‌ی CLIQUE: گواهی در نظر گرفته شده برای این مساله شامل مجموعه رئوسی است که در خوشه قرار دارند؛ برای بررسی درستی آن ابتدا چک می‌کنیم که این مجموعه شامل K عضو متمایز باشد؛ همچنین چک می‌کنیم که هر دو عضو از این K راس به یکدیگر متصل باشند. این کار با $\binom{k}{2}$ انجام خواهد شد که مرتبه‌ی زمانی آن $\mathcal{O}(k^2)$ است. همانطور که دیدیم verify کردن گواهی در زمان چندجمله‌ای انجام شد و در نتیجه این مساله در دسته‌ی NP قرار دارد.

۲) اثبات NP-Hard بودن مساله‌ی CLIQUE: برای اثبات این موضوع، مساله‌ی 3-SAT که می‌دانیم یک مساله‌ی NP-Complete است را به آن کاهش می‌دهیم. برای تبدیل ورودی مساله‌ی 3-SAT به ورودی مساله‌ی CLIQUE به ازای هر literal موجود در هر clause یک راس قرار می‌دهیم. سپس هر دو راس در دو clause متفاوت که با یکدیگر ناسازگار نیستند را با یک یال به یکدیگر متصل می‌کنیم. رئوسی با یکدیگر ناسازگارند که یکی از آنها نقیض دیگری است. برای مثال به ازای ورودی $(x_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor \overline{x}_3)$ گراف ورودی مساله‌ی CLIQUE به صورت زیر خواهد بود:



همانطور که دیده می‌شود x_1 به $\overline{x}_1$، x_2 به $\overline{x}_2$ و x_3 به $\overline{x}_3$ متصل نشده‌اند چون باهم سازگار نیستند. تعداد رئوس و یال‌ها نسبت به ورودی چندجمله‌ای است پس کاهش انجام شده از مرتبه‌ی چندجمله‌ای است. حال ثابت می‌کنیم فرمول 3-SAT، satisfiable است اگر وتنها اگر یک clique یا اندازه‌ی k وجود داشته باشد. ابتدا جهت رفت را اثبات می‌کنیم؛ فرض می‌کنیم که تعداد clause ها برابر با k است. درصورتی که یک جایگذاری satisfiable برای متغیرهای مساله‌ی 3-SAT وجود داشته باشد، در هر clause حداقل یکی از literal ها دارای مقدار true است، راس متناظر با این literal ها را در گراف انتخاب می‌کنیم و می‌دانیم که آنها با یکدیگر سازگارند چون درغیراینصورت مقداردهی

انجام شده در مساله‌ی 3-SAT امکان‌پذیر نبود و از آنجایی که با یکدیگر سازگارند بین آنها یال وجود دارد و یک خوشه با اندازه‌ی k را تشکیل می‌دهد. حال برای اثبات جهت برگشت، اگر خوشه‌ای با اندازه‌ی k وجود داشته باشد، رئوس آن را که در واقع literal ها هستند برابر با true قرار می‌دهیم؛ درصورتی که نقیض یک متغیر باشد، مقدار متغیر برابر با false و درصورتی که برابر با خود متغیر باشد، مقدار آن برابر با true قرار داده می‌شود. از آنجایی که یال‌ها بین رئوسی که با یکدیگر سازگارند قرار داشتند مقداردهی‌های انجام شده با یکدیگر سازگارند و هر راس خوشه باعث شده تا مقدار یکی از clause ها برابر با true شود و درنتیجه فرمول 3-SAT، satisfiable شود.

کاهش مساله‌ی A به مساله‌ی B درشراطی امکان‌پذیر است که ثابت کنیم مساله‌ی A دارای پاسخ است اگر وتنها اگر مساله‌ی B دارای پاسخ باشد؛ توجه کنید که اثبات هر دو جهت شرط ذکر شده در این اثبات ضروری است.

۳. SUBSET-SUM Problem

ورودی مساله‌ی SUBSET-SUM، $S = \{x_1, x_2, \dots, x_n\}$ است که x_1 تا x_n مقادیر صحیح هستند. این مساله بیان می‌کند که آیا می‌توان زیرمجموعه‌ای از مجموعه‌ی S جدا کرد به گونه‌ای مجموع اعضای آن برابر با مقدار صحیح t باشد یا خیر. ثابت کنیم مساله‌ی SUBSET-SUM در دسته مسائل NP-Complete قرار دارد.

پاسخ :

برای اثبات NP-complete بودن این مساله دو گام زیر را انجام می‌دهیم:

۱) اثبات NP بودن مساله‌ی SUBSET-SUM: گواهی در نظر گرفته شده برای این مساله شامل زیرمجموعه‌ای از مجموعه‌ی S است که ادعا می‌شود مجموع اعضایش برابر با t است؛ تعداد اعضای این زیرمجموعه کوچکتر یا مساوی مجموعه‌ی اصلی است پس نسبت به ورودی چندجمله‌ای است. برای بررسی درستی آن چک می‌کنیم که هر یک از اعضای این مجموعه در مجموعه‌ی اصلی وجود داشته باشند و تکراری نیز نباشند که این کار در زمان $\mathcal{O}(n)$ (n تعداد اعضای مجموعه‌ی S است) قابل انجام است؛ سپس مجموع اعضای این زیرمجموعه را محاسبه می‌کنیم تا بررسی کنیم که برابر با t می‌شود یا خیر که این موضوع نیز در زمان چندجمله‌ای قابل انجام است پس SUBSET-SUM عضو دسته‌ی NP است.

۲) اثبات NP-Hard بودن مساله‌ی SUBSET-SUM: برای اثبات این موضوع، مساله‌ی 3-SAT که می‌دانیم یک مساله‌ی NP-Complete است را به آن کاهش می‌دهیم. برای تبدیل ورودی مساله‌ی 3-SAT به ورودی مساله‌ی SUBSET-SUM مراحل زیر را انجام می‌دهیم:

فرض می‌کنیم تعداد clause m مانند C_j و n متغیر مانند x_i داریم. به ازای هر متغیر x_i دو عدد v_i و v'_i که هر یک $n+m$ رقمی است می‌سازیم:

- رقم i ام از دو عدد v_i و v'_i را برابر با ۱ قرار می‌دهیم.
- به ازای $n+1 \leq j \leq n+m$ درصورتی که x_i در C_{j-n} وجود داشت، رقم j ام v_i را برابر با ۱ قرار می‌دهیم.
- بقیه‌ی ارقام اعداد v_i و v'_i را برابر با ۰ قرار می‌دهیم.

برای مثال به ازای ورودی $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$، اعداد v_i و v'_i را به صورت زیر تشکیل می‌دهیم: (توجه کنید که در اینجا n برابر با ۳ و m برابر با ۴ است.)

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	1	0	0	1	0	0	1
v_1'	1	0	0	0	1	1	0
v_2	0	1	0	0	0	0	1
v_2'	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
v_3'	0	0	1	1	1	0	0

همانطور که می‌بینیم ارقام یکم از دو عدد v_1 و v_1' همچنین clause اول را در نظر می‌گیریم و از آنجایی که x_1 در آن قرار دارد، v_1 را برابر با ۱ و v_1' را برابر با ۰ قرار می‌دهیم و از آنجایی که نقیض x_2 و x_3 در آن قرار دارند، v_2' و v_3' را برابر با ۱ و v_2 و v_3 را برابر با ۰ قرار دادیم. به ازای بقیه‌ی متغیرها و clause ها نیز به همین صورت عمل می‌کنیم.
بعد از ساختن اعداد ذکر شده، به ازای هر C_j clause دو slack variable مانند s_j و s_j' که هر دو $n+m$ رقمی هستند را مطابق قوانین زیر می‌سازیم:

- رقم $n+j$ ام از عدد s_j را برابر با ۱ قرار می‌دهیم.
- رقم $n+j$ ام از عدد s_j' را برابر با ۲ قرار می‌دهیم.
- بقیه‌ی ارقام دو عدد s_j و s_j' را برابر با ۰ قرار می‌دهیم.

درنهایت عدد t که یک عدد $n+m$ قرمی است را به عنوان عدد مجموع بدین گونه می‌سازیم:

- به ازای $1 \le j \le n$ رقم j ام از t را برابر با ۱ قرار می‌دهیم.
- به ازای $n+1 \le j \le n+m$ رقم j ام از t را برابر با ۴ قرار می‌دهیم.

اعداد t، s_j و s_j' به ازای مثال گفته شده در زیر نشان داده شده‌اند:

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
s_1	0	0	0	1	0	0	0
s_1'	0	0	0	2	0	0	0
s_2	0	0	0	0	1	0	0
s_2'	0	0	0	0	2	0	0
s_3	0	0	0	0	0	1	0
s_3'	0	0	0	0	0	2	0
s_4	0	0	0	0	0	0	1
s_4'	0	0	0	0	0	0	2
t	1	1	1	4	4	4	4

در نهایت تمامی اعداد ساخته شده به صورت زیر خواهد بود:

- $C_1 = x_1 \lor \overline{x_2} \lor \overline{x_3}$
- $C_2 = \overline{x_1} \lor \overline{x_2} \lor \overline{x_3}$
- $C_3 = \overline{x_1} \lor \overline{x_2} \lor x_3$
- $C_4 = x_1 \lor x_2 \lor x_3$

Two numbers for each variable

Two numbers for each clause

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	1	0	0	1	0	0	1
v_1'	1	0	0	0	1	1	0
v_2	0	1	0	0	0	0	1
v_2'	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
v_3'	0	0	1	1	1	0	0
s_1	0	0	0	1	0	0	0
s_1'	0	0	0	2	0	0	0
s_2	0	0	0	0	1	0	0
s_2'	0	0	0	0	2	0	0
s_3	0	0	0	0	0	1	0
s_3'	0	0	0	0	0	2	0
s_4	0	0	0	0	0	0	1
s_4'	0	0	0	0	0	0	2
t	1	1	1	4	4	4	4

$$\text{Subset} = \{v_1', v_2', v_3, s_1, s_1', s_2', s_3, s_4, s_4'\}$$
$$= \{1000110, 101110, 10011, 1000, 2000, 200, 10, 1, 2\}$$
$$t = 1114444$$

ردیف‌هایی که با رنگ قرمز مشخص شده‌اند تخصیص اعداد ۰ و ۱ به متغیرها را نمایش می‌دهند به گونه‌ای که مجموع هر ستون برابر با رقم t متناظر با آن ستون باشد. n رقم سمت چپ t برابر با ۱ هستند زیرا تنها یکی هر متغیر و نقیضش با یکدیگر ناسازگارند و تنها یکی از آنها می‌تواند برابر با ۱ باشد و بنابراین از بین هر دو ستون v_i و v_i' تنها یکی انتخاب شده است. از آنجایی که برای satisfy کردن هر clause یک، دو و یا هر سه literal موجود در آن برابر با مقدار ۱ است، مجموع مقادیر هر clause حداقل برابر با ۱ و حداکثر برابر با ۳ است پس ارقام مربوط به آن clause در اعداد s_j و s_j' برابر با ۱ و ۲ هستند تا در نهایت مجموع اعداد این ارقام و clause ها برابر با ۴ شود. درواقع درصورتی که مجموع ارقام یک clause برابر با ۱ باشد، هر دو ردیف s_j و s_j' متناظر با آن را انتخاب می‌کنیم تا مجموع ۴ ساخته شود؛ درصورتی که مجموع ارقام یک clause برابر با ۲ باشد، ردیف s_j' متناظر با آن را انتخاب می‌کنیم تا مجموع ۴ ساخته شود و درنهایت درصورتی که مجموع ارقام یک clause برابر با ۳ باشد، ردیف s_j مربوط به آن را انتخاب می‌کنیم تا مجموع ۴ ساخته شود. طبق توضیحات داده شده درصورتی که یک تخصیص satisfiable برای literal های مساله‌ی 3-SAT وجود داشته باشد، مساله‌ی SUBSET-SUM نیز دارای پاسخ است. زیرمجموعه‌ی انتخاب شده به ازای مقال ذکر شده برا با $\{v_1, v_2, v_3, s_1, s_2, s_2', s_3, s_3', s_4\}$ خواهد بود:

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	1	0	0	1	0	0	1
v_2	0	1	0	1	0	1	0
v_3	0	0	1	1	1	0	1
s_1	0	0	0	1	0	0	0
s_2	0	0	0	0	1	0	0
s_2'	0	0	0	0	2	0	0
s_3	0	0	0	0	0	1	0
s_3'	0	0	0	0	0	2	0
s_4'	0	0	0	0	0	0	2
t	1	1	1	4	4	4	4

حال باید نشان دهیم درصورتی که یک زیرمجموعه با مجموع t برای مجموعه وجود داشته باشد، فرمول 3-SAT، satisfiable است؛ برای این منظور درصورتی که v_i در زیرمجموعه باشد، مقدار x_i را برابر با true قرار می‌دهیم و درصورتی که v'_i در زیرمجموعه باشد، مقدار x_i را برابر با false قرار می‌دهیم. به ازای هر متغیر یا خودش و یا نقیضش باید در زیرمجموعه باشد چون در غیراینصورت مجموع آن برابر با ۱ نمی‌شود و حداقل یکی از خود متغیر و یا نقیضش برابر با ۱ است وگرنه مجموع کوچکتر از ۱ خواهد شد و بدین ترتیب همه‌ی clause ها satisty خواهند شد.

۴. مسائل بیشتر

درستی یا نادرستی عبارات زیر را مشخص کنید.

- NP کلاسی از مسائل است که در زمان چندجمله‌ای قابل $verify$ کردن است.
- این موضوع که $P = NP$ است یا خیر تا کنون ناشناخته است.
- اگر مساله‌ای داخل کلاس NP نباشد، در دسته‌ی NP-Complete است.
- اگر مساله‌ای در دسته‌ی NP قرار داشته باشد، در درسته‌ی P نیز قرار دارد.
- اگر مساله‌ای $NP-Complete$ باشد، باید داخل P نیز وجود نداشته باشد.
- مسائل $NP-Complete$ را نمی‌توان به طور موثر تصمیم‌گیری کرد.
- مسائل $NP-Complete$ سخت‌ترین مسائل تصمیم‌گیری هستند.
- فرض کنید $P \neq NP$ و همچنین A و B دو مساله‌ی تصمیم‌گیری هستند؛ اگر A در دسته‌ی $NP-Complete$ باشد و همچنین $A \leq_P B$، در نتیجه A در دسته‌ی P قرار ندارد.
- مساله‌ی تصمیم‌گیری X وجود دارد به طوری که به ازای تمامی Y های داخل NP، Y در زمان چندجمله‌ای قابل کاهش به X است.
- اگر $P = NP$، سپس $P = NP - Complete$.
- اگر مساله‌ای در P نباشد، پس در NP است.
- NP دسته‌ای از مسائل است که در زمان چندجمله‌ای قابل تصمیم‌گیری نیست.