1- راه حل حریصانه: قوی‌ترین و ضغیف‌ترین گاو نر را انتخاب می‌کنیم. اگر این دو گاو می‌توانستند گاوآهن را بکشند، آن‌ها را در یک گروه قرار داده و سپس تعداد گروه‌های ممکن در بقیه گاوها را پیدا می‌کنیم. اما در صورتی که دو گاو انتخاب شده نمی‌توانستند گاوآهن را بکشند، گاو ضعیف‌تر را حذف کرده و تعداد گروه‌های ممکن در گاوهای باقی‌مانده را پیدا می‌کنیم.

این راه حل پاسخ بهینه را به ما می‌دهد. برای اثبات درستی راه حل ابتدا باید 2 لم را اثبات کنیم:

لم 1: اگر قوی‌ترین گاو را s و ضعیف‌ترین گاو را w در نظر بگیریم، اگر s + w < p باشد، w با هیچ گاوی نمی‌تواند در یک گروه قرار بگیرد.

برای اثبات این لم از برهان خلف کمک می‌گیریم. فرض می‌کنیم که w می‌تواند با گاو s′ در یک گروه قرار بگیرد. با توجه به اینکه s قوی‌ترین گاو است، $s' \leq s$ خواهد بود و در نتیجه با توجه به رابطه زیر، w نمی‌تواند با s′ در یک گروه قرار بگیرد و در این حالت به تناقض می‌رسیم، پس w نمی‌تواند در هیچ گروهی قرار بگیرد.

$s' + w \leq s + w < p$

لم 2: اگر قوی‌ترین گاو را s و ضعیف‌ترین گاو را w در نظر بگیریم و فرض کنیم که $s + w \geq p$ است، مجموعه T را مجموعه گروه‌های گاوها در نظر می‌گیریم که می‌توانند گاوآهن را بکشند در حالی که در این مجموعه s و w در یک گروه قرار ندارند. در این حالت، یک مجموعه مانند T′ هم وجود دارد که گاوها را به صورتی گروه‌بندی می‌کند که بتوانند گاوآهن را بکشند و s و w در یک گروه باشند و $|T| \leq |T'|$ باشد.

برای اثبات این لم 2 حالت را در نظر می‌گیریم. در حالت اول حداکثر یک گاو از بین s و w در T هستند و حداقل یکی از این دو گاو در هیچ گروهی نیست. در این حالت T′ را همان T در نظر می‌گیریم با این تفاوت که s و w در هیچ گروهی نباشند. در واقع در این حالت تعداد گروه‌های T′ حداکثر یک واحد کمتر از T خواهد بود. حالا با توجه به اینکه $s + w \geq p$ است، می‌توانیم s و w را با هم در یک گروه قرار داده و به T′ اضافه کنیم. در نتیجه با توجه به اینکه حداکثر یک گروه را از T حذف کردیم اما دقیقا یک گروه اضافه کردیم، تعداد گروه‌های T′ از T کوچک‌تر نخواهد بود یا به عبارتی $|T| \leq |T'|$ است.

در حالت دوم فرض می‌کنیم s و w هر دو در T وجود داشته باشند اما با هم در یک گروه نباشند. در این حالت می‌توانیم فرض کنیم که s با a در یک گروه باشد و w هم با b در یک گروه باشد. حال، با توجه به اینکه w ضعیف‌ترین گاو بوده است و توانسته است با b در یک گروه قرار بگیرد و اینکه می‌دانیم a از w ضعیف‌تر نیست، a و b هم می‌توانند در یک گروه قرار بگیرند. پس در این حالت مجموعه T′ را همان T در نظر می‌گیریم با این تفاوت که گروه s و a و همچنین گروه w و b را حذف کرده و گروه s و w و همچنین گروه a و b را اضافه کرده‌ایم. پس در این حالت تعداد گروه‌های T با T′ برابر است و در نتیجه در این حالت هم لم صحیح است.

حالا به اثبات راه حل حریصانه می‌پردازیم. برای این کار از استقرای قوی استفاده می‌کنیم. فرض می‌کنیم که برای تعداد گاو کمتر از n، راه حل حریصانه بهترین جواب را به ما می‌دهد. حال برای n گاو، اگر پاسخی که راه حل حریصانه به ما می‌دهد مجموعه G باشد و فرض کنیم که G پاسخ بهینه نیست، پاسخ بهینه‌ای برای این تعداد گاو وجود دارد که مجموعه T است. بین این n گاو، گاوی وجود دارد که بیشترین قدرت را دارد که آن را s می‌نامیم و گاوی هم وجود دارد که کمترین قدرت را دارد که آن را w می‌نامیم. اگر $w + s < p$ باشد، طبق لم 1، w در هیچ گروهی نمی‌تواند باشد و در نتیجه در هیچ کدام از مجموعه‌های G و T وجود ندارد.

در این حالت w را از بین گاوها حذف می‌کنیم و می‌دانیم که راه حل حریصانه بهترین جواب را برای n-1 گاو می‌دهد و در نتیجه در این حالت به تناقض می‌رسیم.
در صورتی که $w + s \geq p$ باشد، طبق لم 2 می‌دانیم که مجموعه‌ای مانند T' وجود دارد که همانند راه حل حریصانه s و w را در یک گروه قرار می‌دهد و $|T| \geq |T'|$ است. می‌دانیم که با حذف s و w، راه حل حریصانه یا همان $G - \{(s, w)\}$ پاسخ بهینه را برای n-2 گاو به ما می‌دهد. همچنین می‌دانیم $T' - \{(s, w)\}$ هم راه حل قابل قبولی برای این تعداد گاو است. در نتیجه می‌توان گفت $|T' - (s,w)| \leq |G - (s,w)|$ است و در نتیجه $|T'| - 1 \leq |G| - 1 \rightarrow |T'| \leq |G|$ است. پیش‌تر گفتیم که $|T| \leq |T'|$ است که با کنار هم قرار دادن این دو مورد به نتیجه $|T| \leq |T'| \leq |G|$ می‌رسیم. یعنی در این حالت راه حل حریصانه هم به ما راه حل بهینه‌ای داده است که با فرض تناقض دارد. در نتیجه می‌توان گفت راه حل حریصانه همیشه راه حل بهینه است. برای حالت پایه استقرا هم می‌توانیم حالت $n = 2$ و $n = 3$ را در نظر بگیریم که در هر کدام حداکثر یک گروه خواهیم داشت که بدیهی است راه حل حریصانه پاسخ درست را به ما می‌دهد.
قبل از حل مسئله نیاز است که گاوها را بر اساس قدرتشان مرتب کنیم. مرتبه زمانی انجام این کار $\mathcal{O}(n \log n)$ است. پس از این کار، در هر مرحله یا یک گاو را حذف می‌کنیم و تعداد گروه برای گاوهای باقی‌مانده را محاسبه می‌کنیم و یا اینکه دو گاو را در یک گروه قرار می‌دهیم و تعداد گروه برای گاوهای باقی‌مانده را محاسبه می‌کنیم. در نتیجه حداکثر $n-1$ بار شرط $s_i + s_j \geq p$ بررسی می‌شود، پس می‌توان گفت مرتبه زمانی انجام این کار $\mathcal{O}(n)$ است. در نتیجه مرتبه زمانی کل الگوریتم $\mathcal{O}(n \log n)$ خواهد بود.

2- راه حل حریصانه: کودک با بیشترین اشتها را در نظر می‌گیریم. در صورتی که این کودک با بزرگ‌ترین شیرینی موجود راضی میشد، بزرگ‌ترین شیرینی را به این کودک می‌دهیم و در غیر این صورت کوچک‌ترین شیرینی را به او می‌دهیم و سپس شیرینی‌های باقی‌مانده را بین بقیه کودکان تقسیم می‌کنیم.
این راه حل پاسخ بهینه را به ما می‌دهد. برای اثبات این مورد، بهینه بودن پاسخ را در دو بخش اثبات می‌کنیم.
از این قسمت به بعد فرض می‌کنیم که کودکان بر اساس اشتهایشان و به صورت نزولی و شیرینی‌ها بر اساس سایزشان و به صورت نزولی مرتب شده‌اند.
بخش 1: اگر a_1 را مقدار اشتهای کودک اول (بیشترین اشتها) و s_1 را شیرینی اول (بزرگ‌ترین شیرینی) در نظر بگیریم و $s_1 \geq a_1$ باشد، راه حل بهینه‌ای وجود دارد که این شیرینی را به کودک اول (با بیشترین اشتها) می‌دهد.
برای اثبات این بخش فرض می‌کنیم پاسخ بهینه‌ای مانند O وجود دارد که شیرینی اول را به کودک اول نمی‌دهد. دو حالت را در نظر می‌گیریم، در حالت اول فرض می‌کنیم این پاسخ بهینه شیرینی اول را به هیچ کودکی نمی‌دهد. در این صورت، پاسخ ’O را همان O در نظر می‌گیریم با این تفاوت که شیرینی اول را به کودک اول می‌دهیم. با توجه به اینکه شیرینی بقیه کودکان تغییری نکرده است، تعداد کودکان ناراضی از کودک دوم به بعد ثابت می‌ماند. اما در این حالت می‌دانیم که کودک اول قطعا راضی خواهد بود و در نتیجه تعداد کودکان راضی در پاسخ ’O کمتر از O نخواهد بود.
در حالت دوم فرض می‌کنیم که در پاسخ O، شیرینی اول به کودک i-ام داده شده است و شیرینی j-ام به کودک اول داده شده است. حال پاسخ ’O را همان پاسخ O تعریف می‌کنیم با این تفاوت که شیرینی اول به کودک اول داده می‌شود و شیرینی j-ام به کودک i-ام داده می‌شود. می‌دانیم که در این حالت کودک اول قطعا راضی خواهد بود. همچنین، اگر در پاسخ O کودک اول راضی بوده باشد با توجه به اینکه کودک اول بیشترین اشتها را دارد، کودک i-ام هم راضی خواهد بود. همچنین با توجه به اینکه شیرینی بقیه کودکان تغییری نکرده است، تعداد کودکان راضی در پاسخ ’O از پاسخ O کمتر نخواهد بود. در نتیجه در هر دو حالت، پاسخ بهینه ’O وجود دارد به طوری که بزرگ‌ترین شیرینی را به کودک اول می‌دهد.
بخش 2: اگر $s_1 < a_1$ باشد، راه حل بهینه‌ای وجود دارد که کوچک‌ترین شیرینی را به کودک اول می‌دهد.
برای اثبات این بخش هم فرض می‌کنیم پاسخ بهینه‌ای مانند O وجود دارد که شیرینی m (کوچک‌ترین شیرینی) را به کودک اول نمی‌دهد. دو حالت ممکن است پیش بیاید. در حالت اول، شیرینی m به هیچ کدام از کودکان داده نمی‌شود. در این حالت پاسخ ’O را همان پاسخ O در نظر می‌گیریم با این تفاوت که به کودک اول شیرینی m را می‌دهیم. با توجه به اینکه شیرینی بقیه کودکان تغییری نکرده است، تعداد کودکان راضی از کودک دوم به بعد هم تغییر نکرده است. اما با توجه به اینکه حتی بزرگ‌ترین شیرینی هم نمی‌تواند کودک اول را راضی کند، این کودک در هیچ کدام از دو پاسخ راضی نخواهد بود و در نتیجه تعداد کودکان راضی در هر دو پاسخ یکسان است. در حالت دوم شیرینی m به کودک i-ام داده شده و شیرینی j-ام به کودک اول داده شده است. در این حالت پاسخ ’O را همان پاسخ O در نظر می‌گیریم با این تفاوت که شیرینی m-ام را به کودک اول می‌دهیم و شیرینی j-م را به کودک i-ام می‌دهیم. می‌دانیم که کودک اول در هیچ کدام از پاسخ‌ها راضی نخواهد بود. اما اگر کودک i-ام در پاسخ O راضی بوده باشد، با توجه به اینکه شیرینی j-ام از شیرینی m-ام کوچک‌تر نیست، در پاسخ ’O هم راضی خواهد بود. در نتیجه تعداد کودکان

راضی در پاسخ O' کمتر از این تعداد در پاسخ O نخواهد بود. در نتیجه در هر دو حالت، پاسخ بهینه O' وجود دارد به طوری که شیرینی m-ام را به کودک اول بدهد.

طبق بخش 1 و بخش 2، همواره راه حل بهینه‌ای وجود دارد که به کودک با بیشترین اشتها، شیرینی مطابق با راه حل حریصانه را می‌دهد. به کمک استقرا ثابت می‌کنیم راه حل حریصانه بهینه است. ابتدا فرض می‌کنیم که راه حریصانه بهترین پاسخ برای n-1 کودک را با هر تعداد شیرینی می‌دهد. برای n کودک، فرض می‌کنیم راه حل بهینه O وجود دارد که پاسخی متفاوت با راه حل حریصانه دارد. ابتدا کودک با بیشترین اشتها را کنار می‌گذاریم. می‌دانیم راه حل حریصانه پاسخ بهینه برای n-1 کودک و m-1 شیرینی را دارد (شیرینی مربوط به این کودک حذف شده است). در نتیجه در راه حل O تعداد کودکان راضی بیشتر از راه حل حریصانه نیست. اما با اضافه کردن کودک با بیشترین اشتها، می‌دانیم هر دو راه حل شیرینی یکسانی را به این کودک می‌دهند و رضایت این کودک در هر راه یکسان است. در نتیجه تعداد کودکان راضی در راه حل O بیشتر از راه حل حریصانه نیست و در نتیجه راه حل حریصانه بهینه است.

```
function AssignCookies(s, a, m, n) do
    sortDesc(s)
    sortDesc(a)
    declare ans[n]
    set i, j = 1, m // Assuming that arrays' indexes are 1-based
    for (k = 1; k ⩽ n; ++k) do
        if (s[i] ⩾ a[k]) do
            ans[k] = i
            ++i
        end
        else do
            ans[k] = j
            --j
        end
    end
    return ans
end
```

می‌دانیم مرتبه زمانی مرتب کردن آرایه a، $O(n\,log\,n)$ است و مرتبه زمانی مرتب کردن آرایه s، $O(m\,log\,m)$ است و با توجه به اینکه m > n است، مرتبه زمانی مرتب‌سازی، $O(m\,log\,m)$ خواهد بود. همچنین حلقه موجود در کد n بار اجرا می‌شود که مرتبه زمانی آن $O(n)$ است. در نتیجه مرتبه زمانی کل الگوریتم $O(m\,log\,m)$ خواهد بود.

3- الف) راه حل حریصانه: در هر مکانی که پمپ بنزین وجود دارد، فاصله تا پمپ بنزین بعدی را محاسبه می‌کنیم. اگر این فاصله را می‌توانستیم با بنزین فعلی طی کنیم، در پمپ بنزین توقف نکرده و در غیر این صورت توقف می‌کنیم و باک را پر می‌کنیم.

این راه حل پاسخ بهینه را به ما می‌دهد. برای اثبات درستی راه حل از برهان خلف استفاده می‌کنیم. فرض می‌کنیم که پاسخی که راه حل حریصانه به ما می‌دهد G است و راه حل بهینه‌ای وجود دارد که پاسخ O را به ما می‌دهد و با توجه به فرض خلف، |G| > |O| است. فرض کنیم G به صورت $G = \{g_1, g_2, \dots, g_m\}$ باشد و O به صورت $O = \{o_1, o_2, \dots, o_k\}$ باشد، می‌دانیم که k < m است. اولین i از سمت چپ را پیدا می‌کنیم که $g_i \neq o_i$ است. می‌دانیم پمپ بنزین قبلی در هر دو پاسخ O و G یکسان بوده است یا به عبارت دیگر $o_{i-1} = g_{i-1}$ است (می‌توانیم فرض کنیم باک ماشین در ابتدای مسیر خالی بوده ولی یک پمپ بنزین در مبدا وجود داشته یا به بیان دیگر $o_0 = g_0$ است). با توجه به این مورد و راه حل حریصانه، می‌توان گفت g_i نسبت به o_i در فاصله دورتری از پمپ بنزین قبلی قرار گرفته است زیرا در صورتی که o_i دورتر باشد، طبق راه حل حریصانه، نمی‌توان از پمپ بنزین o_{i-1} به پمپ بنزین o_i رسید. در این حالت پاسخ O′ را همان پاسخ O تعریف می‌کنیم با این تفاوت که o_i و تمام پمپ بنزین‌هایی که بعد از o_i و قبل از g_i قرار دارند را حذف می‌کنیم ($o_i \le o_j < g_i$ تمام o_jها) و به جای آن g_i را قرار می‌دهیم. حال، باید بررسی کنیم که آیا O′ یک پاسخ قابل قبول است یا خیر. در ابتدا می‌دانیم که با توجه به راه حل حریصانه، می‌توانیم از پمپ بنزین o_{i-1} به g_i برسیم. از طرفی، می‌دانیم g_i از همه پمپ بنزین‌های حذف شده نسبت به o_{i-1} دورتر است. به عبارت دیگر، اگر اولین پمپ بنزین موجود در O بعد از پمپ بنزین‌های حذف شده را o_l در نظر بگیریم، می‌دانیم g_i از o_{l-1} نسبت به o_{i-1} دورتر است و در نتیجه قطعا از پمپ بنزین g_i می‌توانیم به پمپ بنزین o_l برسیم. پس می‌توان گفت O′ یک پاسخ قابل قبول است و $|O'| \le |O|$ یا به عبارت دیگر O′ پاسخ بدتری از O نیست و یک پاسخ بهینه است ولی یک قدم به G نزدیک‌تر شده است. اگر |O′| > |O| باشد، با فرض بهینه بودن O متناقض است. اما اگر |O′| = |O| باشد، این کار را ادامه می‌دهیم تا اینکه پاسخ O′ به شکل $O' = \{g_1, g_2, \dots, g_k\}$ برسد. حال، با توجه به راه حل حریصانه و این مورد که k < m است، نمی‌توانیم از g_k به مقصد برسیم که این مورد با فرض قابل قبول بودن O′ و در نتیجه با فرض قابل قبول بودن O تناقض دارد. در نتیجه می‌توان گفت راه حل حریصانه پاسخ بهینه را به ما می‌دهد.

با توجه به اینکه فقط یک بار هر پمپ بنزین را پیمایش می‌کنیم، مرتبه زمانی الگوریتم $O(n)$ است.

ب) راه حل حریصانه: در هر پمپ بنزین اولین پمپ بنزین بعد از پمپ بنزین فعلی که قیمت پایین‌تری از پمپ بنزین فعلی دارد را پیدا می‌کنیم. اگر با یک باک بنزین (پر یا نصفه) می‌توانستیم به این پمپ بنزین برسیم، باک را دقیقا به اندازه‌ای پر می‌کنیم که به این پمپ بنزین برسیم و در پمپ بنزین‌های میانی توقف نمی‌کنیم. در غیر این صورت، باک را به طور کامل پر کرده و در پمپ بنزین‌های بعدی نیز همین کار را تکرار می‌کنیم.

این راه حل پاسخ بهینه را به ما می‌دهد. برای اثبات این راه به روش زیر عمل می‌کنیم:

فرض کنیم پاسخی که راه حل حریصانه به ما می‌دهد $G = \{g_1, g_2, \dots, g_n\}$ باشد که g_i برابر با مقدار بنزینی که در پمپ بنزین i-ام می‌زنیم باشد. اگر راه حریصانه پاسخ بهینه را به ما ندهد، آنگاه پاسخ بهینه‌ای مانند $O = \{o_1, o_2, \dots, o_n\}$ وجود دارد. اولین i را پیدا می‌کنیم به صورتی که $o_i \neq g_i$ باشد. 2 حالت ممکن است رخ دهد:

در حالت اول راه حل حریصانه باک را به طور کامل پر کرده و با توجه به انتخاب حریصانه می‌دانیم حتی با باک پر هم نمی‌توانستیم به یک پمپ بنزین با قیمت کمتر برسیم. در نتیجه پاسخ O باید در یک پمپ بنزین با قیمت بالاتر بنزین بزند که بتواند مسیر را ادامه دهد و با توجه به اینکه هر دو پاسخ قابل قبول هستند، انتخاب پمپ بنزین گران‌تر با فرض بهینه بودن پاسخ تناقض دارد.

در حالت دوم، راه حریصانه باک را به طور کامل پر نمی‌کند که در این حالت دو مورد ممکن است رخ دهد. در مورد اول $o_i > g_i$ است که در این حالت، طبق راه حریصانه می‌دانیم که با مقدار g_i می‌توانیم به پمپ بنزین j-ام برسیم که می‌دانیم $t_j < t_i$ است. در نتیجه، می‌توانستیم مقدار $o_i - g_i$ لیتر بنزین در پمپ بنزین j-ام که قیمت کمتری دارد خریداری کنیم و هزینه کمتری را صرف خرید بنزین کنیم. این مورد با فرض بهینه بودن O متناقض است. در مورد دوم $o_i < g_i$ است. در این حالت با توجه راه حریصانه، مقدار بنزین باک دقیقا برابر با مقداری است که به پمپ بنزین ارزان‌تر از پمپ بنزین فعلی برسیم. در نتیجه حتی اگر یک لیتر کمتر بنزین زده باشیم، به پمپ بنزین ارزان‌تر نمی‌رسیم و مجبور به توقف در پمپ بنزین گران‌تری هستیم که سبب صرف هزینه بیشتری جهت خریداری بنزین می‌شود. این حالت هم با فرض بهینه بودن پاسخ O تناقض دارد. در نتیجه G پاسخ بهینه است.

با توجه به اینکه در این راه هر پمپ بنزین را حداکثر یک بار و در $O(1)$ پیمایش می‌کنیم، مرتبه زمانی الگوریتم $O(n)$ خواهد بود.

4- راه حل حریصانه: ابتدا بازه‌ها را بر اساس زمان شروع و به صورت صعودی مرتب می‌کنیم. سپس در هر مرحله، کوچک‌ترین عددی که در بازه‌های انتخاب شده نیست (اولین زمانی که کتیبه دارای نگهبان نیست) را در نظر می‌گیریم و از بین بازه‌هایی که نقطه شروعشان کوچک‌تر مساوی عددی است که در نظر گرفتیم، بازه‌ای را انتخاب می‌کنیم که دارای بزرگ‌ترین نقطه پایان (بیشترین زمان پایان) باشد. این کار را انقدر تکرار می‌کنیم که بازه مورد نظر به طور کامل پوشش داده شود.

این راه حل پاسخ بهینه را به ما می‌دهد. به منظور اثبات راه، به روش زیر عمل می‌کنیم:

فرض می‌کنیم پاسخی که راه حل حریصانه به ما می‌دهد $G = \{g_1, g_2, \dots, g_m\}$ باشد و این پاسخ، بهینه نباشد. در این صورت پاسخ بهینه‌ای مانند $O = \{o_1, o_2, \dots, o_k\}$ وجود دارد به طوری که |O| < |G| باشد. در این صورت می‌توان یک i پیدا کرد به طوری که $g_i \neq o_i$ باشد. با توجه به نحوه انتخاب حریصانه، می‌دانیم که نقطه پایان o_i کوچک‌تر از نقطه پایان g_i است زیرا راه حل حریصانه بزرگترین نقطه پایان را انتخاب کرده است. در غیر این صورت نقطه‌ای وجود دارد که توسط بازه‌ها پوشش داده نشده است. حال، پاسخ O′ را همان پاسخ O در نظر می‌گیریم با این تفاوت که o_i را با g_i جایگزین کردیم. با توجه به اینکه نقطه پایان g_i دیرتر از نقطه پایان o_i است و همچنین تمام نقاط قبل از این نقطه هم پوشش داده شده است، پاسخ O′ یک پاسخ قابل قبول است. همچنین می‌دانیم این پاسخ بدتر از O نیست اما یک قدم به G نزدیک‌تر شده است. با تکرار این کار به پاسخ $O' = \{g_1, g_2, \dots, g_k\}$ می‌رسیم که یک پاسخ قابل قبول است که بدتر از O نیست (بهینه است). اگر k = m باشد که با فرض بهینه نبودن G تناقض دارد. اما اگر k < m باشد، طبق راه حریصانه می‌دانیم نقطه‌ای در بازه اصلی وجود دارد که پوشش داده نشده است که این مورد با فرض قابل قبول بودن O′ و در نتیجه با فرض قابل قبول بودن O تناقض دارد. در هر حالتی به تناقض می‌رسیم که نشان می‌دهد G یک پاسخ بهینه است.

با توجه به اینکه در ابتدای الگوریتم یک مرتب‌سازی داریم و بقیه الگوریتم فقط یک پیمایش خطی است، مرتبه زمانی انجام الگوریتم $O(n \log n)$ خواهد بود.

5- الف) برای کمینه کردن زمان انتظار صاحبان کارها، در هر مرحله کاری را انتخاب می‌کنیم که زمان انجام کمتری دارد. جهت کمینه کردن زمان اجرا، ابتدا کارها را بر اساس زمان مورد نیاز و به صورت صعودی مرتب می‌کنیم.

برای نشان دادن درستی این الگوریتم فرض می‌کنیم که پاسخی وجود دارد که کار شماره 1 (با کمترین زمان مورد نیاز) را قبل از بقیه کارها انجام نمی‌دهد و انجام این کار را از روز d-ام آغاز می‌کند. در نتیجه کاری مانند w_i $(i \neq 1)$ وجود دارد که قبل از بقیه کارها انجام می‌شود. واضح است که با جابه‌جا کردن این کار و کار اول که زمان انجام کمتری دارد، زمان انتظار برای کارهای بعد از کار w_1 تغییر نمی‌کند ولی زمان انتظار کارهای بین کار w_1 و w_i کمتر می‌شود. همچنین به همان مقدار که زمان انتظار برای کار w_i افزایش پیدا می‌کند، زمان انتظار برای کار w_1 کاهش پیدا می‌کند و در نتیجه این دو مورد همدیگر را خنثی می‌کنند. پس در نهایت، میانگین زمان انتظار برای کارها کاهش پیدا می‌کند.

با توجه به اینکه در ابتدای الگوریتم یک مرتب‌سازی انجام می‌دهیم و باقی مسئله یک پیمایش خطی است، مرتبه زمانی انجام الگوریتم $O(n\,log\,n)$ است.

ب) این مورد هم مشابه مورد الف قابل حل است با این تفاوت که کار با کمترین زمان انجام را زمانی انتخاب می‌کنیم که یا کار قبلی تمام شده باشد و یا اینکه کار جدیدی به ما محول شده باشد. یعنی اگر مشغول انجام کاری باشیم که d_i روز دیگر از آن باقی مانده باشد و کار جدیدی به ما محول شود که d_j روز از ما زمان می‌گیرد به طوری که $d_j < d_i$ باشد، کار قبلی را رها کرده و مشغول انجام کار جدید می‌شویم.

برای انجام این کار می‌توانیم از یک min-heap استفاده کنیم و با اضافه شدن کار جدید، کار را به هیپ اضافه کنیم و بعد از اتمام هر کار یا اضافه شدن کار جدید، مشغول به انجام کاری که در ریشه هیپ قرار دارد شویم (دقت کنیم که با گذشت هر روز باید یک واحد از مقدار ریشه کم کنیم). در نتیجه مرتبه زمانی انجام این الگوریتم نیز $O(n\,log\,n)$ خواهد بود.