ISUCON7 本選マニュアル

スケジュール

本選のスケジュールは以下の時間帯で実施いたします。
運営の都合で競技開始が遅れた場合、競技終了時間も同じだけ遅れます。

- 10:00 競技開始
- 18:00 競技終了
- 以後主催者により結果チェックと追試

参加者は本選開催会場からのみ参加を行い、リモートでの参加は認めません。
ただし、医師による出社停止が認められる症例（インフルエンザなど）に罹患している場合は例外的にリモートでの参加を認めます。

ISUCON7 本選ポータルサイト

(portal URL 割愛)

競技開始後、本選ポータルサイトにログインできるようになります。
事前に共有されたチームID、パスワードを使用してログインしてください。

ポータルサイトでは負荷をかける対象となるサーバーを選択することができます。後述する追試でもこの設定を利用します。追試が実行できない場合は失格になるので、競技終了までにこの情報が正しいことを確認しておいてください。

このページではベンチマーク走行の処理状況も確認できます。緑が走行中のベンチマークジョブを示し、灰色が待機中のジョブを示します。自チームのジョブには * マークが付きます。 ベンチマークが待機中もしくは実行中の間はリクエストは追加できません。

ベンチマークを実行するサーバーと各チームのサーバーはグループ化されています。同一グループの中でベンチマークは同時に1つのみ実行されます。そのため、表示上のキューの順序でベンチマークが実行されるわけではありません。

どのチームが同じグループに属しているかは公開しません。5分以内にベンチマークが始まらない場合はサポートチャットで問い合わせてください。

リーダーボードの更新について

本選ポータル上のリーダーボードのスコアは、ラストの1時間は表示は更新されなくなり、自チームのスコアのみ確認が可能になります。

プロジェクト情報の登録

競技終了後の主催者による確認作業などのため、競技終了までに以下を必ず実施しておいてください。

- ポータル画面の「運営向け情報」から負荷走行を実行するサーバーを選択しておく

時間内に登録作業が行われない場合には失格となりますのでご注意ください。
競技終了後はサーバーに対して一切の操作を行わないでください。運営が追試に利用します。

サーバー、ネットワーク構成について

競技者には4台のサーバーが与えられます。サーバーには連番の名前がついており、初期構成では先頭の3つがWebサーバー、残りの1つがDBサーバーになっています。

各サーバーにはNICが2つ接続されていて、それぞれグローバルIPアドレスとローカルIPアドレスが割り振られています。グローバルIPアドレスはインターネットとの通信で利用します。それに対して、ローカルIPアドレスは用意されている4台のサーバ及びベンチマークサーバーとの通信でのみ利用します。各IPアドレスはそれぞれのサーバーにログインして確認するか、ポータルサイトから確認してください。

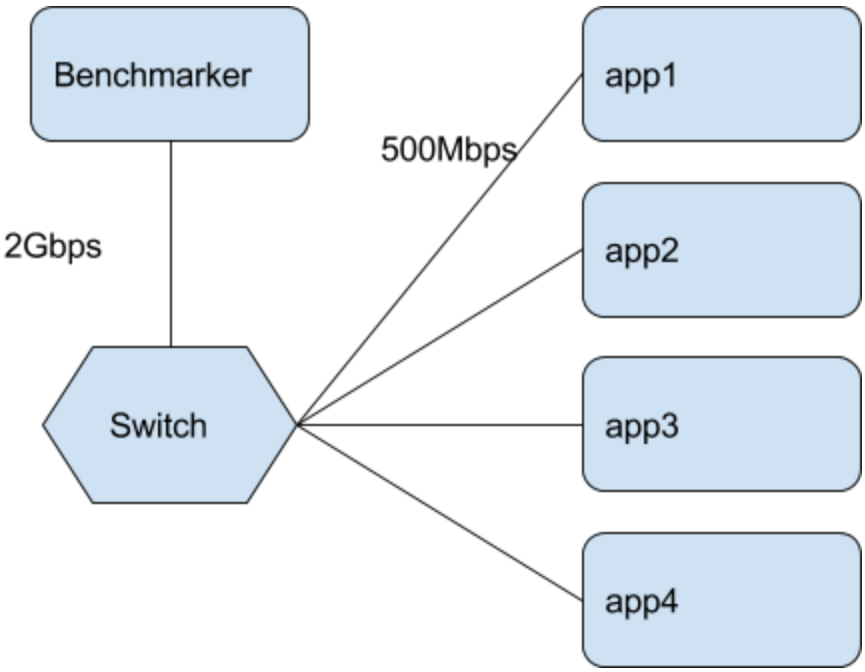
また、各サーバーにはFQDNが割り当てられており、ブラウザなどからFQDNでアクセスすることができます。ブラウザなどからアクセスする場合はDNSにより割り当てられているグローバルIPアドレスに解決されます。

一方ベンチマークを実行する場合、ベンチマークサーバーではブラウザなどと異なりアタック先のFQDNがローカルIPアドレスに解決され、ローカルIPアドレス経由で負荷がかかるようになっています。ですので、Webサーバーがグローバル IPアドレスおよびローカルIPアドレスのどちらのIPアドレスに対しても動作するように（例えば 0.0.0.0:80 を bind するなど）設定してください。

各サーバーの /etc/hosts には、4台のサーバーの短いホスト名とローカルIPアドレスが書かれています。初期構成で、アプリケーションからDBサーバーへの通信はこのホスト名を利用してローカルIPアドレスを使うようになっています。

競技者に与えられる4台のサーバーのローカルIPアドレス側NICと、ベンチマークサーバーは下記の図のように1台のスイッチで接続されています。サーバーとスイッチの間の帯域は、前者が500Mbps、後者が2Gbpsで制限されています。もし、帯域を超えるようなトラフィックが流れたり、一時的に大量の通信（バーストトラフィック）が流れた場合はパケットロスが発生する場合があります。また、下記の図では省略していますが、グローバルIPアドレス側NICは別スイッチと繋がっており、グローバルIPアドレス側の通信はこの500Mbpsの帯域とは別になりま

す。ちなみに、グローバルIPアドレス側の帯域はそれぞれのサーバーで100Mbpsとなっています。



Getting Started

以下の順序で作業を開始してください。

1. サーバーへのログイン

ポータルサイトに書かれてるグローバルIPアドレスまたはホスト名に対して ssh してください。 sshサーバーはパスワード認証になっていて、ユーザー名は isucon 、パスワードはポータルのパスワードと同一です。

サーバーには競技終了後の主催者による確認作業(追試)のため、 ubuntu ユーザーのアカウントがあります。ubuntu ユーザーのアカウントに関して、アカウントの削除や既存の公開鍵の削除を行ったことにより主催者による追試をおこなうことができない場合、失格とします。

2. アプリケーションの動作確認

Webサーバー(グローバルIPアドレスまたはホスト名)にブラウザからアクセスするとアプリが表示されます。

3. 負荷走行

ベンチマーク走行は本選ポータル上からリクエストします。リクエストの際には対象となるサーバーの設定が必要です。初期構成では先頭の3台がWebサーバーになっています。1台以上のWebサーバにチェックを入れて設定を保存してください。

リクエストはキューイングされ、主催者が用意したベンチマーククラスタノードにより順次処理されます。詳細については後述します。

参照実装の切り替え方法

各言語実装は systemd で管理されています。例えば、参照実装をGoに切り替えるには次のようにします。

```
$ sudo systemctl stop    cco.python.service
$ sudo systemctl disable cco.python.service
$ sudo systemctl start  cco.golang.service
$ sudo systemctl enable cco.golang.service
```

PHPを使う場合は systemd の設定変更の他にnginxの設定変更が必要です。

```
$ cd /etc/nginx/sites-enabled
$ sudo unlink ./cco.nginx.conf
$ sudo ln -s ../sites-available/cco.php.nginx.conf .
$ sudo systemctl restart nginx.service
```

MySQLのリカバリ方法

~/db ディレクトリにMySQLのセットアップスクリプトが用意されています。これを使ってデータベースを初期状態に戻すことができます。

```
$ cd ~/db
$ sudo ./init.sh
```

ルール詳細

指定されたサーバー上のアプリケーションのチューニングをおこない、それに対するベンチマークのスコアで競技をおこないます。

与えられたサーバーのみでアプリケーションの動作が可能であればどのような変更を加えても構いません。 ベンチマーク中に与えられたサーバー以外の外部リソースを利用する行為 (他のインスタンスに処理を委譲するなど) は禁止です。ただしモニタリングやテスト、開発などは、PCや外部のサーバーを利用しても構いません。

ベンチマーカーとブラウザの挙動に差異がある場合、ベンチマーカーの挙動を正とします。また初期実装毎に若干の挙動の違いはありますが、ベンチマーカーに影響のない挙動に関しては仕様とします。

ベンチマーク

ベンチマークは以下のように実施されます。

1. 初期化処理の実行 /initialize (制限時間10秒)
2. 事前検証 (10秒程度)
3. 負荷走行 (60秒)
4. 事後検証 (10秒程度)

各ステップで失敗が見つかった場合にはその時点で停止します。ただし、負荷走行中のエラーについては、一部エラーは無視されベンチマークが継続します。

負荷走行中は、毎秒負荷レベル（接続数）が増えていきます。しかし、過去5秒以内に何らかのエラーが発生していた場合は負荷レベルが上昇しません。終了時の負荷レベルはポータルサイトから確認することができます。

スコア計算式を含むベンチマークの詳細についてはこのドキュメントの最後に記載があります。

制約事項

以下の事項に抵触すると失格(fail)となり、点数が無効になります。

- GET /initialize へのレスポンスが10秒以内に戻らない場合
- アプリケーション互換性チェックに失敗した場合
- 他、ベンチマークツールが失敗を検出したケース

最初に呼ばれる初期化処理 GET /initialize は用意された環境内で、チェッカツールが要求する範囲の整合性を担保します。サーバサイドで処理の変更・データ構造の変更などを行う場合、この処理が行っている内容を漏れなく提供してください。またこの処理が10秒以上レスポンスを返さない場合、失格とします。

アプリケーションは全て、保存データを永続化する必要があります。つまり処理実施後に再起動が行われた場合、再起動前に行われた処理内容が再起動後に保存されている必要があります。 また、アプリケーションは、ブラウザ上での表示を初期状態と同様に保つ必要があります。 競技時間終了後に行われる主催者による確認作業(追試)においてこれらの点が確認されます。

アプリケーションについて

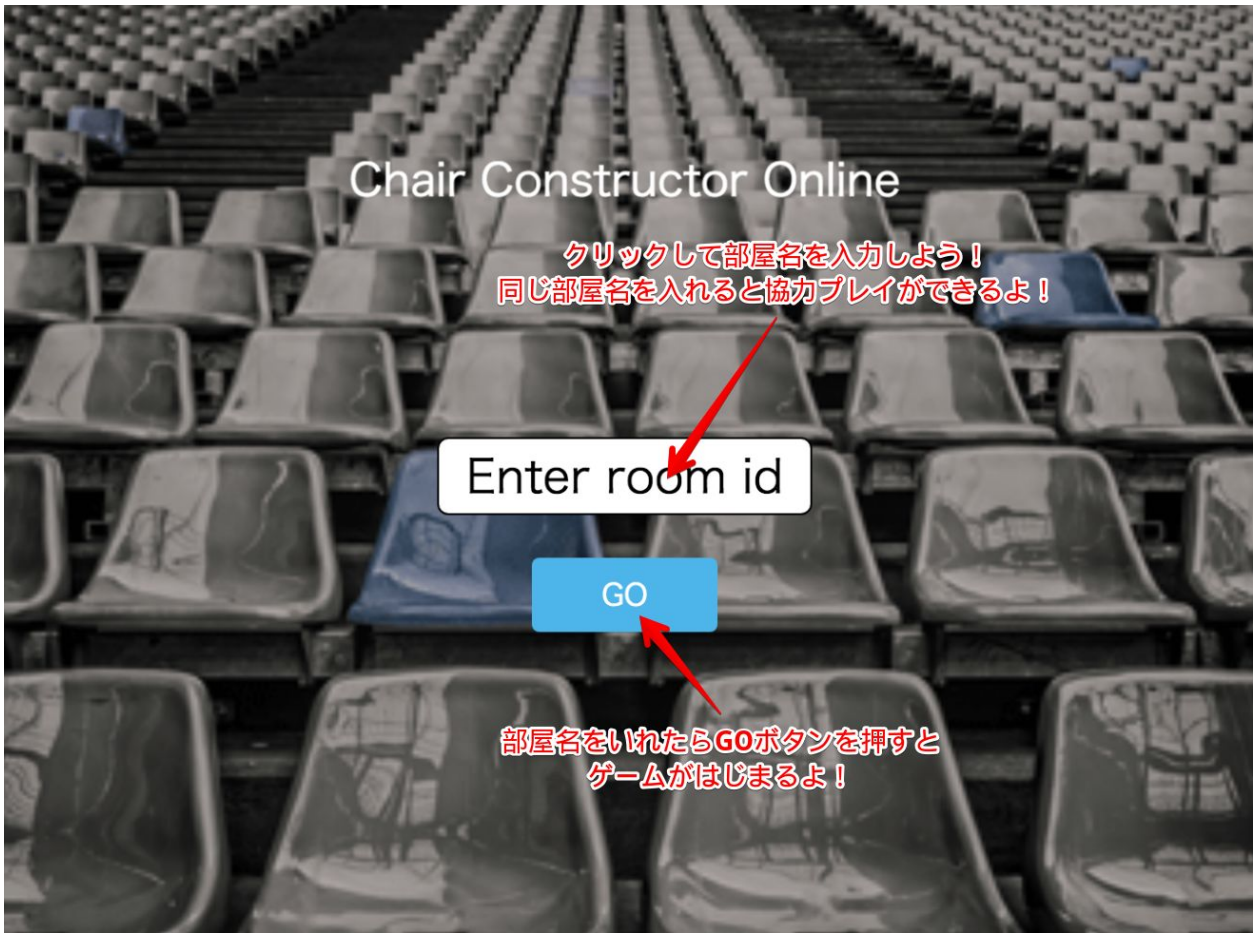
概要

某有名な生産シミュレーションゲームをリスペクトした、マルチプレイヤー・オンラインゲームです。

プレイヤーは椅子をクリックで増やしたり、椅子を生産する設備を購入することができます。

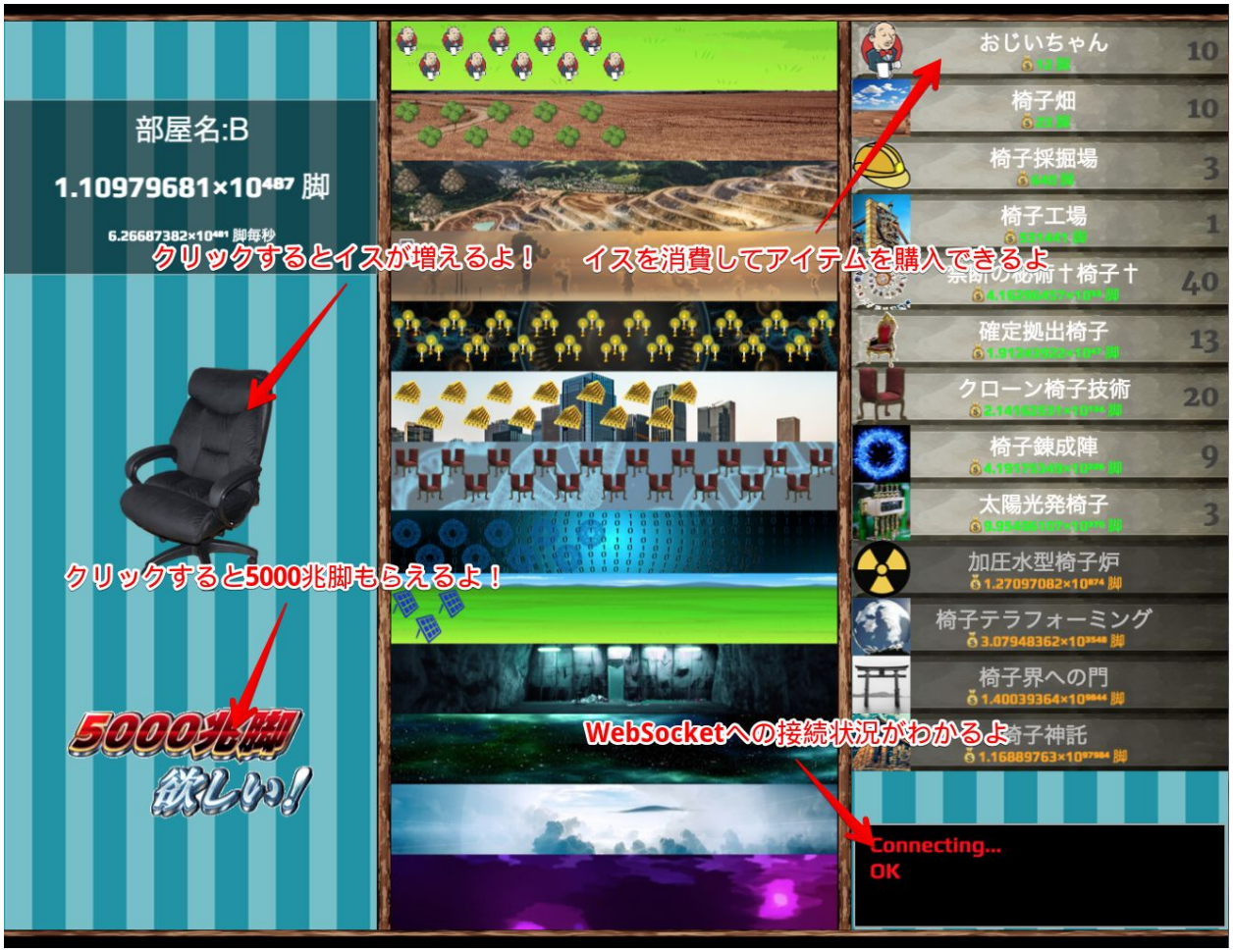
ゲームは「部屋」ごとに分かれていて、同じ部屋に接続しているプレイヤーが協力して椅子を増やします。

サイトのトップページは、各部屋に入るためのポータルサイトになっています。中央のテキストボックスをクリックしてから任意の部屋名を入力し、Goボタンを押すとその部屋のゲーム画面に飛びます。



ゲーム画面では、椅子をクリックすると椅子が増えます。増える数は現在の椅子の生産力に応じて変動します。「5000兆脚欲しい」ボタンを押すと現在の椅子の生産力によらず一気に5000兆脚追加することができます。

画面右には生産設備が並んでいて、クリックすると椅子と引き換えに生産設備を購入することができます。



同期方式

サーバーは現在の状態と、1秒先までの未来の情報の要約をクライアントに定期的に送ります。これをステータスと言います。ステータスにはミリ秒単位のタイムスタンプがついています。

プレイヤーが操作すると、クライアントはコマンドをサーバーに送ります。コマンドにはその操作が反映される、少し未来のタイムスタンプがついています。この未来のタイムスタンプは、ステータスに含まれるサーバーのタイムスタンプに、クライアント上の経過時間を足したものを現在時刻として、更に一定の時間（1秒以下）を足すことで作られます。

ステータスに含まれる schedule[0] の状態は確定した現在の状態です。いずれかのクライアントに返した schedule[0] のタイムスタンプより過去のタイムスタンプに対するコマンドを受け取ったとき、サーバーはそのコマンドにエラーのレスポンスを返さなければなりません。

コマンドの実行に成功したときは、成功のレスポンスを返す前にコマンドの結果を反映したステータスを返さなければなりません。

コマンドとレスポンスの形式は、通信内容やアプリケーションコードから推測してください。ステータスについては複雑なので、次の節で説明します。

ステータス

次のJSONは、「5000兆脚欲しい」ボタンを押したときに返されるステータスの一部です。

```
{
  "time": 1511534554412,
  "schedule": [
    {
      "time": 1511534554406,
      "milli_isu": [0, 0],
      "total_power": [0, 0]
    },
    {
      "time": 1511534555402,
      "milli_isu": [500000000000000, 4],
      "total_power": [0, 0]
    }
  ],
```

トップレベルにある "time" 要素は、サーバー側のなるべく最新のタイムスタンプです。クライアントはこれをもとにコマンドのタイムスタンプを計算します。このタイムスタンプは schedule[0].time と異なっていても構いません。

"schedule[]" は椅子と生産力の時間ごとの状況です。特に schedule[0] は確定した現在の状況を表しています。

"milli_isu" は千分の1個の椅子を表しています。1秒に1つ椅子が生産されるときに、1ミリ秒で生産される椅子の数が「1ミリ椅子」です。
この値は非常に大きくなり得るため、表示に適するように10進数の指数表記をデータ構造に採用しています。具体的には2要素の配列になっていて、1つ目が上位15桁（仮数部）、2つ目はそれに10の何乗をかけるか（仮数部）になっています。たとえば `[500000000000000, 4]` は、$500000000000000 \times 10^4$ という意味です。上位16桁目以降の数字は切り捨てられます。

"total_power" は、購入した生産設備の生産能力（椅子／秒）を、同じく10進指数表記で表したものです。

schedule[n] から schedule[n+1] までの間は、 schedule[n].total_power で椅子の数が線形に増加するものとして扱います。 schedule[0] は常に必要で、それ以外の要素は addIsu や buyItem の効果が発生する時間に存在します。

ステータスのその他の部分の意味については、コード中のコメントや次のページを参照してください。
（当日マニュアル：別紙）

WebSocket接続先の指定方法

アプリケーションは /room/{部屋名} という HTTP API のパスでブラウザやベンチマーカーが WebSocket 接続する先を JSON で返します。JSONの構造は {"host": "ホスト", "path": "パス"} です。

ホストは空文字列でも良く、その場合はそのJSONを返しているサーバーのホストが利用されます。初期実装はこれを利用しています。つまり、ポータルサイトで複数のWebサーバーを指定した場合、ベンチマーカーは通常のHTTPアクセスを指定されたサーバーに分散し、それがそのままWebSocket接続先になります。

クライアントに接続先を指示したい場合は、ここでホストを明示的に返す必要があります。そのときは必ずFQDN (例: "app9901.isu7f.k0y.org") を利用するようにしてください。ローカルIPアドレスを指定してしまうとブラウザから接続できなくなり失格になります。グローバルIPアドレスを指定してしまうと、ベンチマークもグローバルIPアドレスを利用するようになってしまい、性能が出なくなります。

また、ホストを "ホスト名:ポート番号" という形式にして、特定のポートを指定することも可能です。

ベンチマークについて

事前検証

ベンチマーク開始時に基本的な検証を行います。

まずトップページ “/” へアクセスし, HTMLの構造が変更されていないことを確認します。
DOM構造が保たれていれば、空白削除などの変更は許されます。

次に幾つかの静的ファイルを取得して、ファイルが改変されていないことを確認します。
静的ファイルは、1バイトも変更は許されません。
ただし、以下のファイルに対しては改変チェックは行われません。

- debugview.html
- debugview.js

次に幾つかの部屋を作成し、基本的なシナリオを実行して結果を検証します。この段階では予期せぬエラー（接続タイムアウト、レスポンスのタイムアウト、503など過負荷によって起こりうるエラーを含む）は許されません。

負荷走行

事前検証を通過した場合、負荷走行に移ります。

負荷走行では静的ファイルにはアクセスされません。事前検証よりも多くのクライアントが、より複雑なシナリオを実行します。

クライアントは、ステータスやコマンドのレスポンスが1秒以上受信できないときに切断します。

エラーがない場合毎秒部屋が作られ、またすべての部屋でクライアントが増えます。ただし、アクティブな接続数の低下を検出すると、接続数の上限が設定され、それ以上クライアントは増えません。

事後検証

負荷走行が終わると、負荷走行時ログをもとに、計算結果に互換性が保たれているか検証が行われます。検証内容は次の通りです。

- Statusについて
    - scheduleについて
        - timeが昇順であること

        - 時刻 schedule[0].time から 時刻 schedule[0].time + 1000 までの椅子の数と生産速度を正しく表現できていること
    - addingについて
        - 送信した addIsu リクエストに対して整合がとれていること
    - itemsについて
        - m_itemに存在する各item_idに対する要素がちょうど1つずつ存在すること
        - count_built, powerについて
            - schedule[0].time 時点でのその item の数と生産速度を正しく表現できていること
        - count_bought, next_priceについて
            - レスポンス送信時点での buyItem 受理数にもとづいた数と、その数に対応した price であること
        - buildingについて
            - 時刻 schedule[0].time + 1 から 時刻 schedule[0].time + 1000 までの count_built, powerを正しく表現できていること
    - on_saleについて
        - 各要素について
            - 時刻 schedule[0].time の時点で buyItem が可能ならば time は 0 であること
            - buyItem が可能となる時刻が schedule[0].time より大きく schedule[0].time + 1000 以下ならば time はその時刻であること
            - それ以外の場合、 on_sale に要素が存在しないこと
- addIsuおよびbuyItemについて
    - responseが複数存在しないこと
    - 事前検証においてはresponseがちょうど1つ存在すること
    - リクエストとレスポンスのコネクションが同一であること
    - レスポンスの is_success = true のとき、そのリクエストは受理されたものとする
    - 受理されたリクエストについて
        - リクエスト送信より後かつレスポンス受信より前にベンチマークツールが受信したStatusの中でschedule[0].timeが最も大きいStatusについて、リクエスト内容が反映されていること
- buyItemについて
    - (item_id, count_bought) が同一のリクエストについて、複数回受理されていないこと
    - 各 item について、 count_bought の順序が正しいこと

スコア計算

スコアの計算式は以下のとおりです。

スコア = 成功した addIsu リクエスト数 + 10×成功した buyItem リクエスト数

「成功した」とは、 "is_success": true なレスポンスが返されたリクエストのことです。

特別賞

最初に5万点に到達したチームに特別賞が与えられます。

その他

サポートは事前に連絡のあった discord のチャンネルにて行いますが、基本的に、本選環境の構成・操作方法やベンチマークの処理内容については返答しません。

当日マニュアル：別紙

椅子の個数と生産速度について

椅子の生産速度は1秒あたりの生産される個数を表します。

ただし、1ミリ秒単位でゲームを進行させるため、生産速度がN個毎秒のとき、1ミリ秒にNミリ椅子が生産されるとします。

Scheduleやソースコード中では、椅子の個数をミリ椅子単位で表すことがあります。

Status

ある部屋の状態を表します。

Field	Type	Description
time	number	このステータスの送信時刻
adding	Adding配列	1000ミリ秒以内のAdding
schedule	Schedule配列	1000ミリ秒以内のSchedule。要素は時刻順であり、先頭の情報は既に反映済みである必要があります
items	Items配列	1000ミリ秒以内のItems
on_sale	OnSale配列	1000ミリ秒以内のOnSale

Adding

追加された椅子の履歴を表します。

Field	Type	Description

time	number	追加される時刻
isu	string	追加される椅子の個数

Schedule

ある時刻での生産速度等を表現します。

Field	Type	Description
time	number	時刻
milli_isu	Exponential	その時刻での椅子の個数をミリ椅子単位で表したもの。
total_power	Exponential	その時刻で1000ミリ秒間に生産される椅子の個数

Items

各アイテムの状態を表現します。

Field	Type	Description
item_id	number	アイテムのID
count_bought	number	購入された個数
count_built	number	建造された個数
next_price	Exponential	このアイテムを次に購入する場合の価格
power	Exponential	このアイテムの総生産速度

building	Building配列	建造中のこのアイテムの情報

Building

各アイテムの状態を表現します。

Field	Type	Description
time	number	時刻
count_built	number	その時刻でのアイテムの個数
power	Exponential	椅子の生産速度

OnSale

あるアイテムが購入可能になる時刻を表現します。

Field	Type	Description
item_id	number	アイテムのID
time	number	購入可能になる時刻。0は既に購入可能にあることを示します。

Exponential

整数を指数表記で表します。要素を2つ持つ配列であり、先頭の要素が`Mantissa`を、末尾の要素が`Exponent`であるとき、`Mantissa*10^Exponent` を表現します。