

水以下コンテスト 開催の裏側

Kyo_s_s

2024/04/12

自己紹介

自己紹介

- 名前：きょ
- Twitter(現 X): @Kyo_s_s
- 所属：芝浦工業大学 数理科学科 学部 4 年
- AtCoder Algorithm: 青 / Heuristic: 黄
- 第 1 回 / 第 2 回 緑以下コンテスト Writer/Tester
- 水以下コンテストを公式サイト開催しました！
 - 今日はこのことについて話します！



水以下コンテストとは？

水以下コンテストとは？

2024/03/30 に、芝浦工業大学豊洲キャンパスで「水以下コンテスト」を
オンサイト開催しました！

水以下コンテストとは？

2024/03/30 に、芝浦工業大学豊洲キャンパスで「水以下コンテスト」を
オンサイト開催しました！

- 灰 diff 想定 1 問, (だいたい) 水 diff 想定 10 問の計 11 問で, 3 時間
 - 水 diff 想定 10 問にはすべて部分点 1, 2(問題によっては 3 も) が設定
- 黄色以上 : 4pt, 青 : 3pt, 水 : 2pt, 緑以下 : 1pt として,
チームメンバーが 3 人以下かつ合計 4pt 以下でチームが組める

水以下コンテストとは？

2024/03/30 に、芝浦工業大学豊洲キャンパスで「水以下コンテスト」を
オンサイト開催しました！

- 灰 diff 想定 1 問, (だいたい) 水 diff 想定 10 問の計 11 問で, 3 時間
 - 水 diff 想定 10 問にはすべて部分点 1, 2(問題によっては 3 も) が設定
- 黄色以上 : 4pt, 青 : 3pt, 水 : 2pt, 緑以下 : 1pt として,
チームメンバーが 3 人以下かつ合計 4pt 以下でチームが組める

オンサイトには 50 人くらいの方が参加してくれました！

また、オンライン含め全体で約 100 チームの参加がありました！

開催のきっかけ

開催のきっかけ

- くしらっちょさん主催の緑以下コンテストで初めて問題を作問

開催のきっかけ

- くしらっちゃん主催の緑以下コンテストで初めて問題を作問
- 第2回緑以下コンテストがオンラインで開催され、運営として参加
 - 打ち上げで「またオンライン有志コン開催されるといいね」という話になる
 - 開催しよう！

開催のきっかけ

- くしらっちゃん主催の緑以下コンテストで初めて問題を作問
- 第2回緑以下コンテストがオンラインで開催され、運営として参加
 - 打ち上げで「またオンライン有志コン開催されるといいね」という話になる
 - 開催しよう！
- 開催するなら今度はチーム戦がよさそう（緑以下は個人だったため）
 - チーム戦なら ICPC などのように難易度シャッフルしたい
 - (だいたい) 水 diff 想定の問題を集めて、部分点をたくさん設定！

開催のきっかけ

くしらっちょさんの第2回緑以下コンテスト開催記にて ...

開催のきっかけ

くしらっちょさんの第2回緑以下コンテスト開催記にて ...

また、私にオンサイトコンテストというものを教えてくれた Nafmo さん、LT 会の主催をしてくれたユニークビジョンさん、ありがとうございました。

(<https://kusirara.hatenablog.com/entry/2023/12/04/134934> から引用)

開催のきっかけ

ということかというと ...

開催のきっかけ

ということかというと ...

- Nafmo さんがユニークビジョン LT 会で登壇
 - 「緑 Coder が作問をしてオンサイトに 100 人呼んでみた」
<https://nafmo.dev/slides/#uv-study-20230711>

開催のきっかけ

ということかというと ...

- Nafmo さんがユニークビジョン LT 会で登壇
 - 「緑 Coder が作問をしてオンサイトに 100 人呼んでみた」
<https://nafmo.dev/slides/#uv-study-20230711>
 - くしらっちょさんがこの LT を聞いて、「オンサイトいいなあ」となる

開催のきっかけ

ということかというと ...

- Nafmo さんがユニークビジョン LT 会で登壇
 - 「緑 Coder が作問をしてオンサイトに 100 人呼んでみた」
<https://nafmo.dev/slides/#uv-study-20230711>
 - くしらっちょさんがこの LT を聞いて、「オンサイトいいなあ」となる
- くしらっちょさんがオンサイトで第 2 回緑以下コンテストを開催

開催のきっかけ

ということかというと ...

- Nafmo さんがユニークビジョン LT 会で登壇
 - 「緑 Coder が作問をしてオンサイトに 100 人呼んでみた」
<https://nafmo.dev/slides/#uv-study-20230711>
 - くしらっちょさんがこの LT を聞いて、「オンサイトいいなあ」となる
- くしらっちょさんがオンサイトで第 2 回緑以下コンテストを開催
 - 運営として参加し、「ぼくもオンサイト開くぞ！」となる

開催のきっかけ

ということかというと ...

- Nafmo さんがユニークビジョン LT 会で登壇
 - 「緑 Coder が作問をしてオンサイトに 100 人呼んでみた」
<https://nafmo.dev/slides/#uv-study-20230711>
 - くしらっちょさんがこの LT を聞いて、「オンサイトいいなあ」となる
- くしらっちょさんがオンサイトで第 2 回緑以下コンテストを開催
 - 運営として参加し、「ぼくもオンサイト開くぞ！」となる
- 水以下コンテスト開催！

開催のきっかけ

ということかという ...

- Nafmo さんがユニークビジョン LT 会で登壇
 - 「緑 Coder が作問をしてオンサイトに 100 人呼んでみた」
<https://nafmo.dev/slides/#uv-study-20230711>
 - くしらっちょさんがこの LT を聞いて、「オンサイトいいなあ」となる
- くしらっちょさんがオンサイトで第 2 回緑以下コンテストを開催
 - 運営として参加し、「ぼくもオンサイト開くぞ！」となる
- 水以下コンテスト開催！

ユニークビジョン LT 会が無ければ、開催されていなかったかも。
開催していただき、本当にありがとうございます ... ！

作問について

作問について Rime

Rime という作問補助ツールを使用しました.

作問について Rime

Rime という作問補助ツールを使用しました.

Rime とは ... 想定解法・想定誤解法のチェックなどができるツールです.

Test Summary:

subset-mex ... 6 solutions, 29 tests

cpp-correct-all	OK	max 0.06s, acc 0.61s, score 100
py-correct-all	OK	max 0.41s, acc 1.89s, score 100
cpp-correct-task1	OK	all_01.in: Runtime Error, score 20
cpp-correct-task2	OK	all_01.in: Runtime Error, score 60
py-correct-task1	OK	all_01.in: Time Limit Exceeded, score 20
py-correct-task2	OK	all_01.in: Runtime Error, score 60

作問について Rime

Rime という作問補助ツールを使用しました。

Rime とは ... 想定解法・想定誤解法のチェックなどができるツールです。

Test Summary:

subset-mex ... 6 solutions, 29 tests

cpp-correct-all OK max 0.06s, acc 0.61s, score 100

py-correct-all OK max 0.41s, acc 1.89s, score 100

cpp-correct-task1 OK all_01.in: Runtime Error, score 20

cpp-correct-task2 OK all_01.in: Runtime Error, score 60

py-correct-task1 OK all_01.in: Time Limit Exceeded, score 20

py-correct-task2 OK all_01.in: Runtime Error, score 60

「この愚直実装は部分点1ではACし、部分点2では落ちる」のようなことを確認する必要があり、すべての問題で Rime を使用しました。

作問について Rime

C++ と Python の両方で AC できることを確認しながらテストケースを作成していたのですが ...

作問について Rime

C++ と Python の両方で AC できることを確認しながらテストケースを作成していたのですが ...

- C++ では十分高速に AC できるが, Python だと TL がぎりぎり / TLE
- Python で余裕をもって AC できるように制約を調整したら, C++ の本来落としたい解法が通ってしまう
 - 部分点 1 のみ通ってほしい C++ 解法が, 部分点 2 の制約で高速に通ってしまう ... 等

ということが発生し, 部分点の制約設定が意外と大変でした.

作問について Rime

C++ と Python の両方で AC できることを確認しながらテストケースを作成していたのですが ...

- C++ では十分高速に AC できるが, Python だと TL がぎりぎり / TLE
- Python で余裕をもって AC できるように制約を調整したら, C++ の本来落としたい解法が通ってしまう
 - 部分点 1 のみ通ってほしい C++ 解法が, 部分点 2 の制約で高速に通ってしまう ... 等

ということが発生し, 部分点の制約設定が意外と大変でした.

Rime によるコマンド 1 つでテストできるため, 効率的に作業できました!

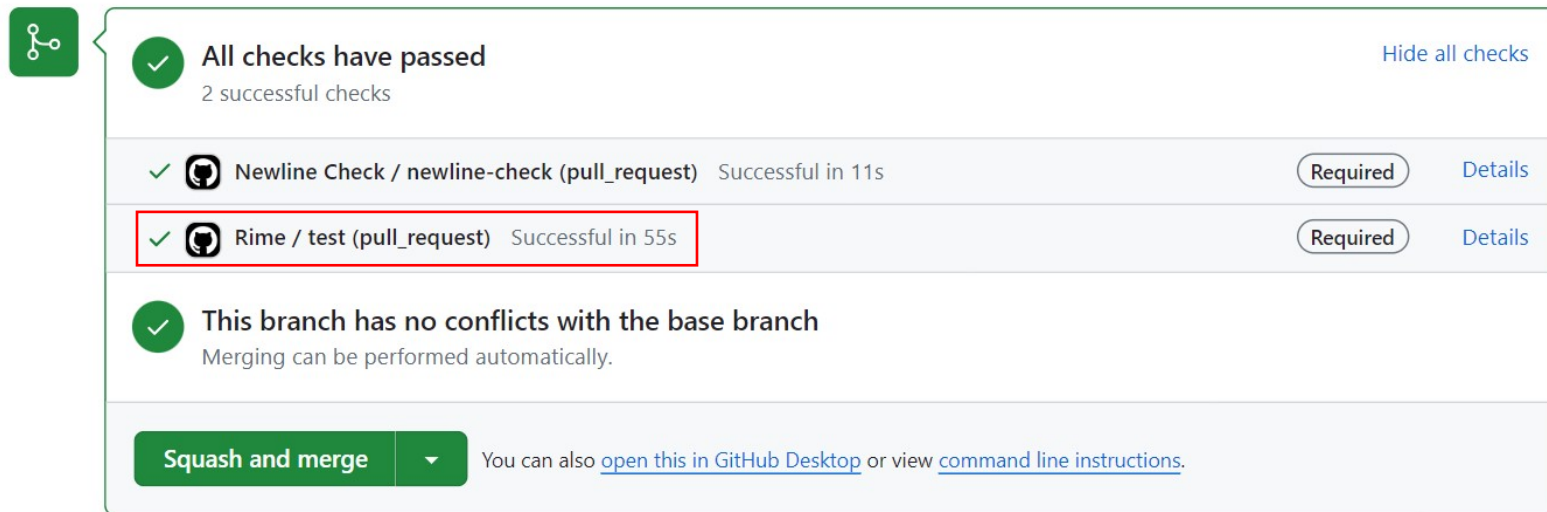
作問について GitHub

問題はすべて GitHub で管理しました.

作問について GitHub

問題はすべて GitHub で管理しました。

GitHub Actions を用いてテストを自動化し、Rime のテストを通過しないと Merge できないように設定したりしました。



The screenshot displays a GitHub Actions workflow status interface. At the top left is a green square icon with a white branching diagram. To its right is a green circle with a white checkmark, followed by the text "All checks have passed" and "2 successful checks". A blue link "Hide all checks" is on the far right. Below this are three rows of check status:

- Row 1: A green checkmark, the GitHub logo, the text "Newline Check / newline-check (pull_request)", "Successful in 11s", a "Required" badge, and a "Details" link.
- Row 2: A green checkmark, the GitHub logo, the text "Rime / test (pull_request)", "Successful in 55s", a "Required" badge, and a "Details" link. This row is highlighted with a red rectangular border.
- Row 3: A green checkmark, followed by the text "This branch has no conflicts with the base branch" and "Merging can be performed automatically."

At the bottom, there is a green button labeled "Squash and merge" with a dropdown arrow, followed by the text "You can also [open this in GitHub Desktop](#) or view [command line instructions](#)."

作問について GitHub

Rime のテストを通過しないと Merge できないように設定したため、常に main ブランチはテストを通過している状態を保つことができました。

作問について GitHub

Rime のテストを通過しないと Merge できないように設定したため、常に main ブランチはテストを通過している状態を保つことができました。

→ 心理的な安心を保ちつつ作業できました。

作問について GitHub

Rime のテストを通過しないと Merge できないように設定したため、常に main ブランチはテストを通過している状態を保つことができました。

→ 心理的な安心を保ちつつ作業できました。

インターン先で GitHub Actions を使ってテストを自動化していて、「水以下コンでも同じようにテスト自動化したい！」と思い、初めて GitHub Actions を書きました。

作問について GitHub

Rime のテストを通過しないと Merge できないように設定したため、常に main ブランチはテストを通過している状態を保つことができました。

→ 心理的な安心を保ちつつ作業できました。

インターン先で GitHub Actions を使ってテストを自動化していて、「水以下コンでも同じようにテスト自動化したい！」と思い、初めて GitHub Actions を書きました。

→ 業務プログラミングは競技プログラミングの役に立つ！(?)

おわりに

おわりに 今日のまとめ

- 水以下コンテストをオンサイト開催したよ！
 - きっかけを辿るとユニークビジョンさんの LT 会のおかげ！
本当にありがとうございます！！

おわりに 今日のまとめ

- 水以下コンテストをオンサイト開催したよ！
 - きっかけを辿るとユニークビジョンさんの LT 会のおかげ！本当にありがとうございます！！
- 作問には Rime を使用し，GitHub で管理したよ！
 - 部分点をたくさん設定していたけど，Rime を活用できた！
 - GitHub Actions を初めて使ってテストを自動化した！安心！
 - 業プロは競プロの役に立つ！（？）

おわりに

まだまだ話せなかった裏話がたくさんあります.

- GitHub Actions の無料枠を使い切りそうになってしまい、慌てて高速化した話
- 当日 MOFE に気軽に寄付できるようにした話
- 原案として出されたが、解けなくて没になった問題の話
- 難易度推定はやっぱり難しかった話
- PyPy が Python より断然速かった話
- スライド・名札をすべて SATySF_I で書いた話
- ...

気になるある話があれば、是非この後話しかけてください！

ご清聴ありがとうございました！
