木幅アルゴリズムの学習システムの構築

進 捗 状 況

B4 小林紹子

October 27, 2025

目次

研究背景

- 木幅はグラフ構造理論などアルゴリズム設計で重要な概念.
- しかし,
  - 理解が難しい（抽象的な概念, 木分解の構築の難しさ）
  - 日本語で学べる教材や可視化・インタラクティブな教材が少ない.
- ⇒「木幅アルゴリズムを直感的に学べる教材」が求められている.

研究目的

- 木幅や木分解の理解を支援する学習サイトを開発する.
- 対象：グラフ理論をある程度学習している人（例：情報系学生）.
- 目的：
  - 定義の理解促進.
  - アルゴリズムの可視化による直感的理解.

要件定義 I

1. 問題管理機能（サーバー側）
   - 問題文, 画像, 正答を含む問題データを保持.
   - API 経由でデータのやり取りを行う.
2. 問題表示機能（クライアント側）
   - サーバーから問題リストを取得して表示.
   - 問題文と対応する画像の表示.
   - ユーザーが解答を選択・入力可能.

要件定義 II

3. 解答送信・結果表示機能
   - 解答送信時にサーバーにリクエストを送信.
   - サーバー側で判定し, 結果を返す.
   - ページ遷移せずに結果を即時表示.
4. インタラクティブに図をマウス等で操作.
   - 操作し変更された図の頂点をデータとして保持.
   - 解答として送信可能.
5. 相手の解答パターンに合わせた正解を自動生成.
   - それ以前に選択された答えによって異なる解答が得られる.
   - サーバー側でインタラクティブに生成.

将来的な拡張要件

- 管理者画面での問題追加・削除.
- ユーザー登録.
- 回答履歴・正答率の記録.

このまま説明する前に

- これから使用技術について説明したい.
- 事前に説明してみたら, 井口先生にウェブプログラミングを履修していない学生が多いと言われた.
- 今後のためにもまずは基本的な Web アプリケーションについての仕組みを説明します...

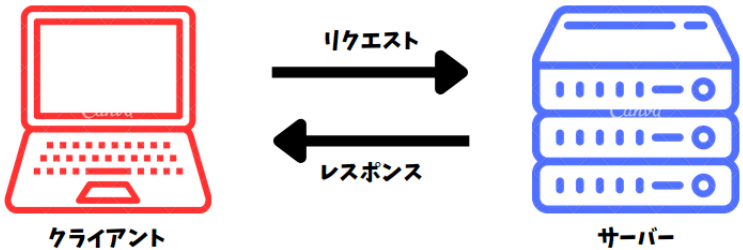
クライアントとサーバー

- **クライアント**：サーバーに指示を出すデバイス（例：PC, スマートフォン）[1].
- **サーバー**：
  サービスや機能を提供するコンピュータ. クライアントから要求された形に情報を加工したり, 保存したりする役割がある. サーバーにも役割によって種類がある.
  - Web サーバー：Web 関連のサービスを提供しているコンピュータ.
  - メールサーバー：メール関連のサービスを提供しているコンピュータ.
  - DNS サーバー：URL を IP アドレスに変換するコンピュータ.

[1] サーバーとは？ クライアントとは？ それぞれの違い https://menter.jp/blog/server-and-client

リクエストとレスポンス

- Web サイトを閲覧する際にクライアントからサーバーに通信することを「**リクエスト**」と呼ぶ.
- サーバーがリクエストを受け, クライアントに返答することを「**レスポンス**」と呼ぶ.



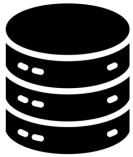
Web アプリケーションと Web サイトの違い

- **Web サイト**：常に同じ HTML をクライアントに返す.
- **Web アプリケーション**：
  表示内容をクライアントごとに動的に変化する HTML を返す.

1. Web ブラウザから URL を入力して Enter.
   リクエスト情報が Web アプリに送信（リクエスト）.
2. リクエストが Web アプリに届くと, リソース（HTML, CSS）をデータベースと連携して取得.
3. ブラウザに返す（レスポンス）.

データベース

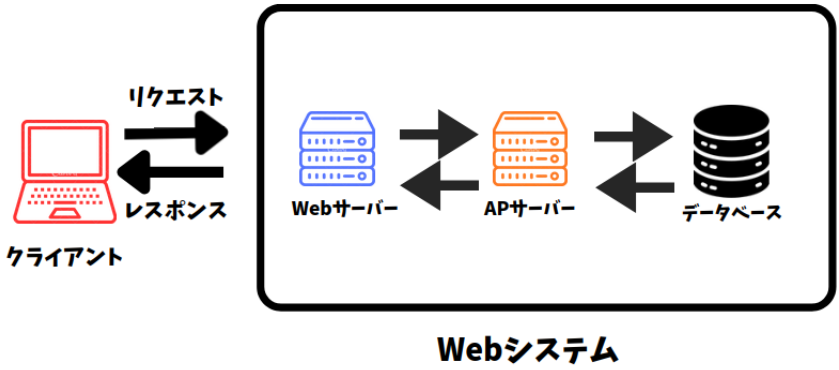
- **データベース**：
  - 構造化した情報, データの組織的な集合 [2].
  - 通常はコンピュータ・システムに電子的に格納.
  - **データベース管理システム（DBMS）**で制御.



- **データベース管理システム（Database Management System）**：
  - データベースとそのエンドユーザー・プログラムの間でインタフェースとして機能.
  - データベースの監視・制御を容易にする.
  - 例）MySQL

[2] データベースとは www.oracle.com/jp/database/what-is-database/

Webアプリケーションの流れ



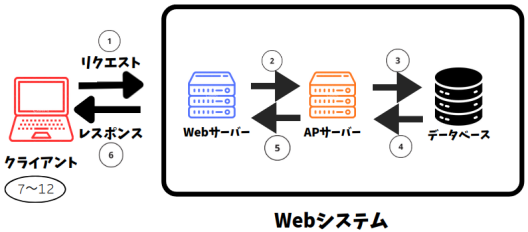
Web サーバーと AP サーバー

- **Web サーバー**：
  - HTTP リクエストを受け取る [3].
  - HTML や CSS, 画像などでレスポンスを返す.
- **AP サーバー (Web アプリケーションサーバー)**：
  - Web サーバから受け取った情報を処理 [4].
  - 必要に応じてデータベースサーバにアクセス.
  - 動的コンテンツの生成・管理.

Web サーバはユーザーとの窓口.
AP サーバはユーザーのリクエストに応じた対応をおこなうバックヤード.

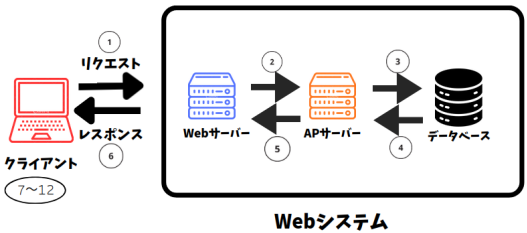
Webサイトがブラウザに表示されるまでの大まかな流れ！

1. Web サーバーにアクセスしてリクエストを送信.
2. Web サーバーがリクエストを受け取り, 解析.
3. AP サーバーが処理を実行.（DB 問い合わせなど）



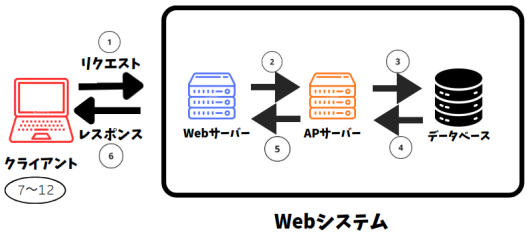
Webサイトがブラウザに表示されるまでの大まかな流れ II

4. DB から情報を取得.
5. 静的な HTML と Javascript コード or JSON などのデータを Web サーバーに返す.
6. Web サーバーがレスポンスとしてクライアントに送信.



Webサイトがブラウザに表示されるまでの大まかな流れ III

7. ブラウザ（クライアント）が HTML を解析.DOM ツリーを構築.
8. CSS を読み込み, 解析.CSSOM ツリーを構築.
9. Javascript を読み込み実行. 動的な処理（データ取得・DOM 操作など）を行う.

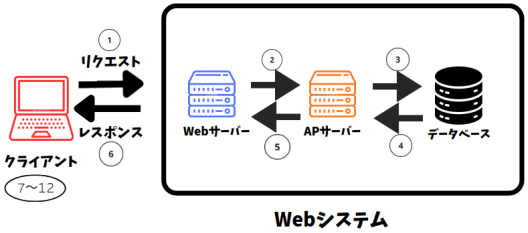


Webサイトがブラウザに表示されるまでの大まかな流れ IV

10 DOM ツリーと CSSOM ツリーを合成してレンダーツリーを構築.

11 レイアウト計算・ペイント.

12 画面にレンダリング.



SSR（Server Side Rendering）

サーバーが返す HTML にもどの程度の完成度にするか違いがある.

- **SSR（Server Side Rendering）**
  - 登場：2000 年代前半〜中盤.（PHP, Ruby on Rails など）
  - ユーザーがページを開く度にサーバーが HTML を「最初から全部生成」して返す.
  - クリックなどの遷移も毎回サーバーにリクエスト.
  - ブラウザは HTML を受け取って DOM ツリーを作る.
- **特徴**
  - 表示が遅い（ページ全体を毎回描きなおす）.
  - SEO（検索エンジン対策）には強い.

React

- 2011 年 Facebook 社（現 Meta 社）のソフトエンジニアが React の前進となるプロトタイプを開発.2013 年にオープンソース化.
- 画面の見た目を管理するための仕組みをライブラリ化したもの.
- HTML を直接書かず,「状態に応じて UI を自動で作る」仕組みを持つ.
- 特徴
  - 宣言的 View：あらかじめ変更される部分を {} で囲むなど変更されない部分と区別して設計.
  - コンポーネントベース：部品を組み立てるように作成する. 容易＋再利用可能になる.
  - 仮想 DOM：実際の DOM を直接操作せず, メモリ上にツリー構造を作り, 変化部分だけを検出して更新する.

SPA（Single Page Application）

- **SPA（Single Page Application）**
  - 登場：2010 年代前半〜中盤.（React, Angular.Vue など）
  - サーバーから空っぽの HTML を受け取る.
  - ブラウザが JS を読み込み, そこから DOM ツリーと仮想 DOM ツリーを生成.
  - ページ遷移は JS が行う（サーバーに行かずにクライアント側で完結）.
  - 仮想 DOM ツリーの更新前と更新後を比べ, そこだけ書き換える.
- **特徴**
  - ページ再読み込みがないためサクサク動く.
  - 初期表示が遅め（JS が全部読み込まれないと表示できない）.
  - SEO に弱い.

SSR + React（モダン SSR）

- **SSR + React（モダン SSR）**
  - 登場：2017 年頃.（React v16 以降（Next.js など））
  - サーバーが React コンポーネントを実行して HTML 生成を行い, クライアントに送る.
  - ブラウザが HTML を表示.
  - 表示後,React の JS が実行. **ハイドレーション**して HTML に動きをつける.
- **特徴**
  - 初期表示が速い（HTML が最初に出る）.
  - SPA のように動的.
  - ハイドレーションで「静的 HTML→動的操作可能」に.

ハイドレーション（Hydration）

- サーバーサイドで生成された HTML は静的な HTML.
- ボタンを押そうが入力があろうが何も起きない.
- クライアント側で HTML と一緒に送られた JS コードから作った DOM ツリーを再接続することでイベントが可能.
- この再接続することを**ハイドレーション**という.

RSC（React Server Components）

- **RSC（React Server Components）**
  - 登場：2023 年頃.（Next.js +13）
  - React コンポーネントのうち, 一部をサーバーで実行.
  - →今まで React コンポーネントはクライアント側で動く Javascript 関数だった.
  - クライアント側は一部の UI だけを JS で動かす.
  - サーバーでデータ取得や重い処理を行い,HTML 部分だけ送信.
- **特徴**
  - データ取得が高速（直接サーバーで DB アクセス）.
  - JS 転送量が減る（クライアントは最小限）.
  - SSR よりも効率的・柔軟.

使用技術（現段階）

- Node.js：JavaScript をサーバーで実行できるようにした環境.
- Next.js：React を使って Web アプリケーションを構築するためのフレームワーク.Node.js 上で動作.
- React：UI を作るためのライブラリ.
- React Flow：インタラクティブなダイアグラムを簡単に実装できる.

環境構築

- npm をインストール（sudo apt npm）version：v22.15.0
- Next.js をインストール（npx create-next-app@latest –ts next-sample）version：v22.15.0
- ここで Turbopack が開発サーバークラッシュしてるとエラーがでた.
- →nodemodule と package.json を削除
- npm i で再インストールするとうまくいった.
- npm run dev で初期画面がでてきた.（http://localhost:3000）
- npm install reactflow
- 作れはしたが、もともとチャートフロウなどを書く用に作られたものだったので上手な図形は作れず.

○×問題

- まずはバックエンド・フロントエンドに問題・回答を書いた.
- JSON で答えをサーバからもらって判断する.
- 将来的にはデータベースから問題を取得するようにする.

次回までに

- データベースの作成：SQLite を使用する予定.
- React Flow の調整：上手くいかなかった場合は代替えの技術の調査.

参考文献 I

[1] MENTER 編集者. サーバーとは？ クライアントとは？ それぞれの違い. https://menter.jp/blog/server-and-client, (参照日 = 2025/10/22).

[2] データベースとは. https://www.oracle.com/jp/database/what-is-database/, (参照日 = 2025/10/22).

[3] Web アプリケーションサーバとは？ web サーバとの違いや 3 層構造についても解説. https://dx.nid.co.jp/column/what-is-a-web-application-server, (参照日 = 2025/10/22).

[4] Ap サーバーとは? https://cmc-japan.co.jp/blog/ap-server/, (参照日 = 2025/10/22).