Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

INFORMATYKA

lekcja pokazowa/otwarta –
szkoła ponadpodstawowa

Scenariusz lekcji

Temat:

Strategia dziel i zwyciężaj w porządkowaniu przez scalanie

Maciej M. Sysło
syslo@ii.uni.wroc.pl
http://mmsyslo.pl

Wrocław miasto spotkań
DOLNY ŚLĄSK
Dolnośląska Biblioteka Pedagogiczna we Wrocławiu
Uniwersytet Wrocławski
GMINA DŁUGOŁĘKA
Wrocławskie Centrum Doskonalenia Nauczycieli
Dolnośląska Szkoła Ćwiczeń

1. Obszar informatyki i klasa, w której będą prowadzone zajęcia

Obszar informatyki: Algorytmika i programowanie – szkoła podstawowa.

Klasa: wybór klasy zależy od nauczyciela, który powinien wziąć pod uwagę, czy uczniowie są przygotowani do zajęć na zaproponowany temat.

Przygotowanie uczniów do lekcji: uczniowie wiedzą, na czym polega problem porządkowania (powinni znać ze szkoły podstawowej), znają przynajmniej jedną metodę porządkowania (w szkole podstawowej poznali porządkowanie przez wybór); programują w języku Python lub C++, rozumieją rekurencję i potrafią zastosować funkcją rekurencyjną (poznali w szkole podstawowej przy rysowaniu).

2. Zagadnienia metodyczne stanowiące podstawę przygotowania lekcji w zakresie rozwijania kompetencji metodycznych

Metodyczną podstawę realizacji lekcji na zaplanowany temat, a faktycznie każdej lekcji realizującej obowiązującą podstawę programową informatyki, stanowią fundamenty kształcenia informatycznego w polskich szkołach, wynikające z konstrukcji podstawy programowej, miejsca i roli programowania, powiązań informatyki z innymi dziedzinami oraz metodyki kształcenia. Tymi fundamentami są:

1) **kolejność** celów ogólnych kształcenia – kształcenie w zakresie abstrakcyjnego i algorytmicznego myślenia powinno poprzedzać naukę programowania i korzystanie z aplikacji komputerowych;
2) **spiralność** kształcenia wokół tych samych informatycznych pojęć, metod i celów przez wszystkie lata w szkole od pierwszej klasy po ostatnią klasę;
3) **myślenie komputacyjne** – jednym z głównych celów kształcenia informatycznego jest rozwój sposobów myślenia angażowanego w formułowanie problemu i przedstawianie rozwiązań w taki sposób, aby komputer – człowiek lub maszyna – mógł skutecznie je wykonać; jest to sedno podejścia informatycznego do rozwiązywania problemów;
4) **programowanie** – etap kreatywnego rozwiązywania problemów z wykorzystaniem komputera – *learning by doing,* konstrukcjonizm;
5) **informatyka w swoich zastosowaniach** – nauczanie przez rozwiązywanie problemów z różnych dziedzin;
6) **metoda projektów** – zalecana w podstawach wszystkich przedmiotów, praca w zespołach – informatyk nie pracuje dzisiaj sam, samodzielnie, tylko w zespole.

Te ogólne wskazania zostały uwzględnione w tej propozycji scenariusza i powinny zostać uwzględnione i zrealizowane przez nauczyciela, który ma poprowadzić pokazową/otwartą lekcję według tego scenariusza.

Porządkowanie informacji, w szczególności liczb, zwane też sortowaniem, jest jedną z najważniejszych i najczęściej wykonywanych operacji w komputerze. Algorytmy porządkowania należą do klasyki informatyki i kształcenia informatycznego na każdym poziomie edukacji. Ta lekcja (**spiralnie**) następuje wtedy, gdy uczniowie znają już problem porządkowania i wcześniej nauczyli się porządkować liczby metodą przez wybór – było to w szkole podstawowej.

Na tej lekcją mają okazję poznać kolejną metodę porządkowania, bazującą na technice **dziel i zwyciężaj**, która polega na dekompozycji problemu i redukcji do mniejszych instancji tego samego problemu oraz użyciu rekurencji. Dodatkowo jest stosowana operacja scalania, a sam algorytm nosi nazwę **porządkowania przez scalanie**. W procesie prowadzącym do poznania tej metody stosowane są więc podstawowe metody myślenia komputacyjnego: abstrakcja, dekompozycja i redukcja oraz rekurencja.

W porządkowaniu przez scalanie najpierw ciąg, który ma być uporządkowany, jest dzielony (**krok dekompozycji** i **redukcji**) na dwa w miarę równoliczne podciągi, następnie te dwa podciągi są porządkowane (**rekurencyjnie**) … tą samą metodą, i już uporządkowane są **scalane** w jeden ciąg uporządkowany.

Zgodnie z zalecaną metodyką uczniowie najpierw poznają odpowiednie algorytmy – scalania i podziału problemu na mniejsze oraz użycia rekurencji – a następnie, gdy rozumieją, na czym polegają te algorytmy, o czym mogą się przekonać wykonując je odręcznie na papierze, przystępują do ich programowania dla komputera. Na końcu testują poprawność programów i ewentualnie poprawiają znalezione błędy.

3. Temat lekcji: Strategia dziel i zwyciężaj w porządkowaniu przez scalanie

4. Treści nauczania

Treści nauczania na tej lekcji odnoszą się bezpośrednio do następujących zapisów w podstawie programowej informatyki dla szkół ponadpodstawowych (liceum i technikum)

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach oraz algorytmy:
   e) **sortowania ciągu liczb przez scalanie**,
3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:
   b) **rekurencję** ([…], **sortowania liczb**, […]),
   c) **metodę dziel i zwyciężaj** ([…], **sortowanie przez scalanie** […]),

Pojęcia i metody informatyczne, którym są poświęcone zajęcia:

- podejście algorytmiczne dziel i zwyciężaj,
- scalanie ciągów uporządkowanych w ciąg uporządkowany,
- porządkowanie ciągu przez scalanie,
- rekurencja,
- funkcja rekurencyjna,
- komputerowa realizacja porządkowania przez scalanie.

5. Cele/osiągnięcia ucznia

Uczeń:

- stosuje dekompozycję problemu na mniejsze,
- scala dwa ciągi uporządkowane w jeden ciąg uporządkowany,
- stosuje rekurencję w porządkowaniu,
- porządkuje ciąg, stosując metodę przez scalanie,
- tworzy model wywołań rekurencyjnych,
- programuje funkcję i stosuje ją w programie rekurencyjnym,
- uruchamia swój program w Pythonie i ewentualnie poprawia go, gdy działa niezgodnie ze specyfikacją,
- testuje działanie programu na różnych danych.

6. Metody pracy z uczniem, formy pracy uczniów

- **Słowna i Oglądowa**: dyskusja w klasie, moderowana przez nauczyciela, ma doprowadzić do ogólnego sformułowania algorytm scalania, a później sortowania przez scalanie z wykorzystaniem rekurencji. Nauczyciel posługuje się talią kart i odpowiednimi planszami, na których umieszcza schematy algorytmów i obliczeń.
- **Czynna**: uczniowie najpierw dyskutują z nauczycielem i między sobą nad rozwiązaniem postawionego problemu, później indywidualnie ćwiczą z użyciem kart i testują poznane algorytmy odręcznie na wielu przykładach, a na końcu piszą odpowiedni program w języku Python, debagują go i testują na tych samych przykładach, które stosowali w obliczeniach odręcznych, programowanie może przebiegać w parach.
- **Aktywizująca**: nauczyciel może przydzielać uczniom wcześniej przygotowane zestawy danych dla algorytmów i programów, które mogą służyć do głębszego prześledzenia działania algorytmów; uczniowie mogą aktywizować się nawzajem, np. pracując w parach – programowanie w parach jest powszechnym podejściem przy tworzeniu oprogramowania, pojawia się również w szkołach; w parach uczniowie mogą się aktywizować między sobą, nawzajem.

7. Środki dydaktyczne wykorzystywane przez nauczyciela i przez uczniów

Do przybliżenia uczniom algorytmów scalania i porządkowania przez scalanie nauczycielowi może być przydatna **talia kart** do gry (z naturalnym porządkiem kart kolorami i figurami) lub zestaw kart z liczbami. Podobnie zademonstrowane przez nauczyciela algorytmy uczniowie mogą w pierwszej kolejności zastosować właśnie do zestawów kart, by lepiej poznać te algorytmy przed przystąpieniem do ich wykonywania na papierze, a później programowania. Nauczyciel może poprosić na poprzedniej lekcji, by uczniowie również przynieśli na zajęcia talie kart.

Do demonstracji działania porządkowania przez scalanie nauczyciel i uczniowie mogą posłużyć się aplikacją w sieci pod adresem https://www.toptal.com/developers/sorting-algorithms – należy wybrać kolumnę Merge, w której z kolei znajdują się cztery rodzaje danych do uporządkowania (te dane to odcinki porządkowane co do długości, od najkrótszego u góry).

W załączeniu jest również krótka prezentacja dotycząca rekurencji – do przypomnienia, co to jest rekurencja (slajdy 2-4) i do wyjaśnienia mechanizmy rekurencji w porządkowaniu przez scalanie (slajdy (5-7).

Już nie jako środek dydaktyczny ale jako wzorcowe rozwiązania nauczyciel może posłużyć się gotowymi programami, realizującymi omawiane algorytmy. Może je znaleźć np. w punkcie 10.2 w książce M.M. Sysło *Algorytmy* [4], jak również w [5].

8. Przebieg lekcji – Lekcja odbywa się w pracowni komputerowej

Wprowadzenie do lekcji

Wprowadzenie do lekcji może najpierw polegać na przypomnieniu, czym jest porządkowanie i przypomnienie algorytmu porządkowania przez wybór, który uczniowie znają. Uczniowie mogą przy tym posłużyć się aplikacjami dydaktycznymi: aplikacją pod linkiem https://www.toptal.com/developers/sorting-algorithms (należy wybrać *Selection*) i programem[1] *Maszyna sortująca* (w załączeniu).

W ramach wprowadzenia nauczyciel może również poświęcić chwilę na przypomnienie uczniom pojęcia rekurencji. Posłużyć do tego mogą 3 pierwsze slajdy załączonej prezentacji.

Zasadnicza część lekcji

Nauczyciel przedstawia, na czym polega porządkowanie przez scalanie. Posługuje się przy tym talią kart i schematami obliczeń. Uczniowie również będą posługiwali się taliami kart, wykonując algorytmy, które wcześniej objaśni nauczyciel.

Algorytm porządkowania przez scalanie może być dobrą ilustracją dwóch procesów związanych z myśleniem komputacyjnym: myślenia abstrakcyjnego przy dekompozycji problemu na takie same problemy o mniejszych rozmiarach i łączeniu rozwiązań tych podproblemów w rozwiązanie problemu głównego.

Na rysunku jest przedstawiamy ideowy schemat porządkowania przez scalanie.

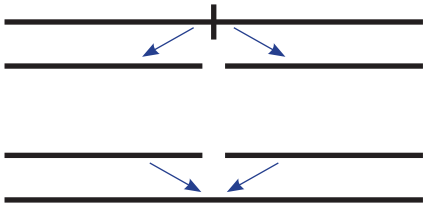
Dany ciąg liczb do uporządkowania:

Podział ciągu na dwie części:

Porządkowanie obu podciągów (**tą samą metodą**) – **rekurencja**

Podciągi uporządkowane

Scalenie podciągów uporządkowanych w jeden ciąg



Nauczyciel ilustruje ten schemat za pomocą talii kart, nie wchodząc jednak w szczegóły. Zaczyna od sterty kart. Dzieli ją na dwie w miarę równe sterty i układa obok siebie. Następnie twierdzi, że te dwie stery zostały już uporządkowane tą samą metodą i na końcu scala je w jedną stertę, dobierając po jednej karcie z jednej lub drugiej uporządkowanej sterty.

[1] Aplikacja ta pochodzi z podręcznika: E. Gurbiel, G. Hardt-Olejniczak, E. Kołczyk, H. Krupicka, M.M. Sysło, *Informatyka*, WSiP, Warszawa 2009.

Z tego ogólnego opisu wynika, że do utworzenia algorytmu porządkowania przez scalanie należy wiedzieć:

a) jak dzielić porządkowany ciąg na dwa podciągi,
b) jak scalać dwa uporządkowane ciągi w jeden ciąg uporządkowany,
c) jak wywoływać zagłębiające się etapy tego procesu podziału i scalania,
d) kiedy kończą się wywołania rekurencyjne.

Odpowiedzi na te wszystkie punkty, z wyjątkiem b), są dość oczywiste: a) – ciąg jest dzielony na prawie połowy, c) – ten sam program jest wywoływany dla podciągów, to jest podejście rekurencyjne, d) – wywołania dla podciągu kończą się, gdy nie można go podzielić na dwa podciągi, czyli gdy składa się z jednego elementu.

Pozostaje do wyjaśnienia, jak scalić dwa uporządkowane podciągi w jeden ciąg uporządkowany. Można to wykonać, przeglądając oba podciągi od najmniejszych elementów i przenosząc do tworzonego ciągu ten z widocznych elementów, który jest nie większy – ilustrujemy to poniżej:

Nauczyciel ilustruje ten algorytm scalania ciągów uporządkowanych za pomocą kart – oczywiście scalane sterty powinny zawierać uporządkowane karty.

Indywidualna praca uczniów. Uczniowie wykonują algorytm scalanie, posługując się własnymi taliami kart. Zamiast scalania talii kart mogą ten algorytm wykonać w zeszycie: piszą nad sobą dwa uporządkowane ciągi liczb, mogą być różnej długości, i nadpisują nad liczbami numery, w jakiej kolejności te ciągi będą scalane.

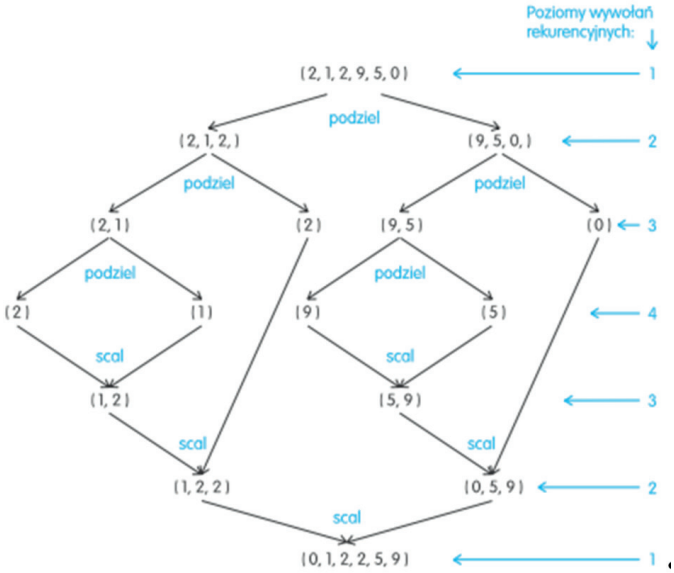
Uwaga. Dobrze jest pokazać, że przebieg algorytmu scalania nie zależy od liczby kart w stertach. Można na przykład wziąć jedną stertę uporządkowanych kart, a w drugiej umieścić kartę o największej liczbie. *Pytanie do uczniów*: w jakiej kolejności będą opróżniane sterty?

Wyzwanie dla uczniów. Nauczyciel stawia pytanie: ile porównań (przeniesień kart) należy wykonać, aby scalić dwa uporządkowane ciągi odpowiednio o k i l elementach? Uczniowie liczą, posługując się swoimi przykładami.

Podpowiedź. W najgorszym wypadku jest to $k + l - 1$, i ta liczba nie zależy ani od wartości k, ani od wartości l, ale od tego, jakie są porządkowane liczby. Najgorszy przypadek to taki, gdy w obu stertach przed końcem scalania pozostaje po jednej karcie.

Zasadnicza część lekcji – wyprowadzenie algorytmu rekurencyjnego

Dobrze jest posłużyć się ilustracją przebiegu algorytmu poniżej, jest też na slajdzie nr 5, ale nie na slajdzie nr 6.
Nauczyciel demonstruje pełny algorytm z użyciem talii nieuporządkowanych kart.
Prosi też ucznia-ochotnika to powtórzenia tego schematu na tablicy, ale dla innego ciągu.



Następnie każdy z uczniów dla wybranego przez siebie nieuporządkowanego ciągu rysuje podobny schemat u siebie w zeszycie. Nie wystarczy posłużyć się talią kart, można się zgubić.

Programowanie – tutaj w języku Python

Ten etap lekcji następuje w momencie, gdy wszyscy uczniowie poprawnie wykonali jeden i drugi algorytm na taliach kart i/lub odręcznie w zeszycie. Dzięki temu postępują z zasadą

zanim zaczniesz programować,
wpierw pomyśl komputacyjnie.

Chociaż cały algorytm porządkowania wydaje się dość złożony, to okazuje się, że zaprogramowanie scalania podciągów wymaga większej uwagi, niż napisanie głównego programu rekurencyjnego.

Sugestie dla nauczyciela

a) uczniowie mogą programować w parach – to obecnie jedno z popularniejszych podejść metodycznych w nauce programowania, uczniowie wtedy pomagają sobie i motywują się;
b) na slajdzie nr 7 jest pokazany schemat rekurencyjnego programu porządkowania przez scalanie – można go udostępnić uczniom jako zarys programu;
c) tym, którzy mają kłopoty z zaprogramowaniem jednego z algorytmów lub obu, można udostępnić teksty obu programów z książki M.M. Sysło, *Algorytmy*, p. 10.2.

Kolejnym bardzo ważnym etapem w programowaniu jest testowanie, czy program jest poprawny i ewentualnie debagowanie, czyli poprawianie znalezionych błędów. Uczniowie powinni testować swoje programy na przykładach, na których wykonywali algorytmy w zeszytach.

Dodatkowe wyzwanie dla uczniów – test na poprawne rozumienie działania rekurencji i na myślenie rekurencyjne

Pytanie, w jakiej kolejności są wykonywane operacje podziału i scalania w powyższym schemacie algorytmu porządkowania przez scalanie?

Podpowiedź. Nie jest to kolejność poziomami od góry, a później od dołu, czyli najpierw dzielenie ciągu na podciągi, a później scalanie, jak może to sugerować powyższa ilustracja.

Wskazówka. Proponujemy dokładnie, wielokrotnie prześledzić w aplikacji, w jaki sposób jest wykonywany ten algorytm: https://www.toptal.com/developers/sorting-algorithms,

I jaki stąd wniosek? Jeśli nadal trudno go sformułować, to nauczyciel wyświetla slajd nr 6 z załączonej prezentacji. Demonstrowana jest na nim dokładna kolejność operacji w procedurze rekurencyjnego porządkowania przez scalanie.

Ocena pracy uczniów

Ocenianie powinno być motywujące uczniów kolejno przydzielanymi zadaniami do wykonania odręcznie na etapie poznawania algorytmów. Zadania są przydzielane wszystkim uczniom, ale zakres ich wykonania może być różny, może wymagać pomocy innego ucznia lub wsparcia przez nauczyciela.

Ocena postępów w programowaniu powinna uwzględniać zakres samodzielności w programowaniu, ale o tym uczniowie powinni być informowani na każdych zajęciach informatycznych. Taki system oceniania może dopingować uczniów do podejmowania coraz odważniejszych prób samodzielnego programowania, bez pomocy ucznia w parze, bez wsparcia nauczyciela i bez korzystania z pomocy „z sieci", czyli korzystania z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.

Uwaga. Niestety, zmorą zadań programistycznych (np. domowych) jest obecnie nagminne korzystanie z kodów dostępnych w sieci, gdzie znajdują się niezliczone ilości kodów w każdym języku programowania dla każdego zadania. Nauczyciel powinien to wziąć pod uwagę i wypracować własny system oceniania zadań programistycznych.

Ewentualna modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Kontynuacja zajęć poszerzających zakres porządkowania przez scalanie może polegać na policzeniu, ile porównań jest wykonywanych w całym tym algorytmie, patrz p. 10.2 w [4]. Ponadto tę metodę porządkowania można porównać z innymi algorytmami porządkowania, np. pod względem liczby wykonywanych porównań.